Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Μεταγλωττιστές

Παραδείγματα Ενοτήτων 7-9

Ενότητα 7: Ενδιάμεσος κώδικας

Άσκηση 7-1:

Θεωρήστε τη γλώσσα προγραμματισμού C με τη γνωστή γραμματική και σημασιολογία, καθώς και το παρακάτω πρόγραμμα αυτής:

```
int x,z;
void A (int * i) {
   static int x = 0;
   x = x + *i;
   *i = z + x;
}
void main() {
   int y = 0;
   x = z = 1;
   A(&x);
   if (x > y) A(&z);
   else {
      int j = x;
      A(&j);
      z = j;
   }
}
```

Να δώσετε μια μορφή ενδιάμεσου κώδικα σε μορφή αφηρημένου συντακτικού δέντρου για το παραπάνω πρόγραμμα. Συγκεκριμένα, δώστε (α) τον πίνακα δέσμευσης μεταβλητών για κάθε μονάδα του προγράμματος, και (β) το αντίστοιχο δέντρο ενδιάμεσου κώδικα, αν υπάρχει. Σε κάθε πίνακα δέσμευσης αποθηκεύονται οι θέσεις των μεταβλητών στο εγγράφημα δραστηριοποίησης ή στο χώρο στατικών δεδομένων, οι βασικοί τύποι αυτών και πιθανά οι αρχικές τους τιμές. Κάθε εμφάνιση μεταβλητής σε ένα δέντρο συνοδεύεται από την ένδειξη του πίνακα στον οποίο αυτή δεσμεύεται και τον αντίστοιχο δείκτη σε αυτόν.

Απάντηση

Σε ένα πρόγραμμα της γλώσσας C όλες οι εντολές βρίσκονται μέσα σε συναρτήσεις, συμπεριλαμβανομένης της συνάρτησης του κυρίως προγράμματος που έχει το όνομα “main”. Όλες οι συναρτήσεις δηλώνονται στο ίδιο επίπεδο εμβέλειας, χωρίς δηλαδή να επιτρέπονται φωλιασμένες δηλώσεις, κάτι που γίνεται για παράδειγμα στην Pascal.
Το επίπεδο εμβέλειας στο οποίο δηλώνονται οι συναρτήσεις περιλαμβάνει και δηλώσεις καθολικών τύπων και μεταβλητών. Αν δεν υπάρχουν κι άλλα αρχεία που να συμμετέχουν στη μετάφραση του προγράμματος, αυτό είναι το πιο εξωτερικό επίπεδο εμβέλειας. Από τις δηλώσεις του επιπέδου αυτού θεωρούμε ότι μόνο οι δηλώσεις μεταβλητών συμμετέχουν στον ενδιάμεσο κώδικα, και μάλιστα μόνο στην κατασκευή του πίνακα δέσμευσης των στατικών δεδομένων, όχι δηλαδή στην κατασκευή εκτελέσιμου κώδικα.
Ο χώρος στατικών δεδομένων δεν περιλαμβάνει μόνο τις μεταβλητές του εξωτερικού επιπέδου εμβέλειας του προγράμματος, καθώς η τοποθέτηση μιας μεταβλητής στο χώρο αυτό δε σχετίζεται με την εμβέλειά της, παρά μόνο με τη διάρκεια ζωής της. Κάθε μεταβλητή του

προγράμματος, η οποία έχει διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος, τοποθετείται στο χώρο στατικών δεδομένων. Εκτός λοιπόν από τις καθολικές μεταβλητές, και οι στατικές μεταβλητές της C, μεταβλητές δηλαδή που είναι δηλωμένες με το χαρακτηρισμό "static", ανήκουν σε αυτό το χώρο, σε οποιαδήποτε μονάδα κι αν είναι δηλωμένες.
Πιο συγκεκριμένα, επειδή μια στατική μεταβλητή έχει διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος, η θέση της στο χώρο δεδομένων πρέπει να είναι σταθερή. Έτσι, όταν το πρόγραμμα εκτελείται, κάθε φορά που ο έλεγχος ροής οδηγεί στη συνάρτηση στην οποία αυτή είναι δηλωμένη, η στατική μεταβλητή θα προσπελαύνεται στην ίδια θέση, και η τιμή της θα διατηρείται μεταξύ διαδοχικών εισόδων στον κώδικα της μονάδας. Παρ' όλο που η μεταβλητή έχει εμβέλεια μόνο εκείνη τη συνάρτηση, κι επομένως όταν η μεταγλώττιση αυτής ολοκληρώνεται, η μεταβλητή διαγράφεται από τον πίνακα συμβόλων, μαζί με τις υπόλοιπες μεταβλητές της ίδιας εμβέλειας, η δέσμευση του χώρου δεδομένων για τη μεταβλητή αυτή διατηρείται μέχρι το τέλος της μεταγλώττισης του προγράμματος, γι' αυτό και δε μπορεί παρά να γίνεται στο χώρο στατικών δεδομένων.
Αντίθετα με τις στατικές μεταβλητές, οι υπόλοιπες τοπικές μεταβλητές μιας συνάρτησης – όπως και οι τυπικές παράμετροι της συνάρτησης – έχουν διάρκεια ζωής μόνο μία κλήση της συνάρτησης, και δε μπορούν να διατηρήσουν την τιμή τους μεταξύ κλήσεων. Η δέσμευση χώρου δεδομένων γι' αυτές γίνεται στο εγγράφημα δραστηριοποίησης της μονάδας, και κατά την εκτέλεση του προγράμματος τοποθετούνται στη στοίβα, γι' αυτό και ονομάζονται μεταβλητές στοίβας. Η θέση τους τυπικά μεταβάλλεται από κλήση σε κλήση της συνάρτησης, αν στον κώδικα του προγράμματος η συνάρτηση καλείται από πολλές διαφορετικές θέσεις, ή αν συμμετέχει σε αναδρομή. Τότε, την επόμενη φορά που η ροή ελέγχου οδηγεί στη μονάδα, το εγγράφημα δραστηριοποίησης της μονάδας μπορεί να βρίσκεται σε διαφορετικό σημείο στη στοίβα από το σημείο όπου βρισκόταν την προηγούμενη.

Η αρχικοποίηση μιας μεταβλητής στη γλώσσα C γίνεται στην αρχή της διάρκειας ζωής της. Από τα παραπάνω προκύπτει ότι μια αρχικοποίηση μπορεί να γίνει με έναν από τους εξής δύο τρόπους:

- Αν η μεταβλητή είναι καθολική ή στατική, η αρχική της τιμή τοποθετείται στον πίνακα δέσμευσης, και από εκεί μπορεί να αποθηκευτεί απ' ευθείας στο αρχείο τελικού κώδικα που θα παραχθεί από το μεταγλωττιστή. Αυτό συμβαίνει, επειδή η αρχική τιμή δίνεται μία μόνο φορά, στην αρχή εκτέλεσης του προγράμματος.
- Αν η μεταβλητή είναι μεταβλητή στοίβας, ο μεταγλωττιστής οφείλει να κατασκευάσει κώδικα που θα της δίνει την αντίστοιχη αρχική τιμή κάθε φορά που η ροή ελέγχου οδηγεί στη μονάδα όπου αυτή δηλώνεται, χειριζόμενος ουσιαστικά την αρχικοποίηση σαν έκφραση ανάθεσης.

Μπορούμε να θεωρήσουμε ότι ο χώρος στατικών δεδομένων της C ορίζει μια μονάδα, την πιο εξωτερική μονάδα του προγράμματος, στην οποία όμως δεν υπάρχουν εντολές. Έτσι, ο ενδιάμεσος κώδικας αυτής της μονάδας θα είναι μόνο ο πίνακας δέσμευσης του χώρου στατικών δεδομένων. Ο πίνακας αυτός δεν παράγει εκτελέσιμο κώδικα, αλλά συμμετέχει στην παραγωγή τελικού κώδικα για το υπόλοιπο πρόγραμμα, καθορίζοντας επακριβώς (α) τη θέση και το μέγεθος του δεδομένου που θα προσπελαστεί στη μνήμη με κάθε αναφορά στην αντίστοιχη μεταβλητή από τον κώδικα, και (β) την αρχική του τιμή, αν αυτή είναι ορισμένη.
Για κάθε άλλη μονάδα πέρα από την παραπάνω εικονική μονάδα, ο χώρος δεδομένων πρέπει εκτός από το χώρο των τοπικών μεταβλητών, να συμπεριλαμβάνει χώρο για τις παραμέτρους της αντίστοιχης συνάρτησης, για την τιμή που αυτή επιστρέφει, αν δεν είναι τύπου void, αλλά και για αποθήκευση της διεύθυνσης επανόδου από τη συνάρτηση. Μπορούμε να θεωρήσουμε αυθαίρετα[1] ότι η διεύθυνση επανόδου, έστω τύπου void*, αποθηκεύεται στη θέση 1, ότι η τιμή αποτελέσματος, αν υπάρχει, αποθηκεύεται στη θέση 2, και ότι οι παράμετροι αποθηκεύονται σε διαδοχικές θέσεις από τη θέση 3 του χώρου δεδομένων της μονάδας. Στη συνέχεια ακολουθούν οι υπόλοιπες μεταβλητές στοίβας της συνάρτησης. Ο μεταγλωττιστής είναι πι-

[1] Συνήθως η αυθαιρεσία περιορίζεται από συμβάσεις που υπαγορεύουν οι μεγάλοι κατασκευαστές μεταγλωττιστών και αρχιτεκτονικών για λόγους φορητότητας και συμβατότητας λογισμικού.

θανό να δεσμεύσει πρόσθετες θέσεις μετά από τις προηγούμενες για αποθήκευση προσωρινών μεταβλητών (όπως για παράδειγμα για διάχυση καταχωρητών). Κάτι τέτοιο συμβαίνει μετά την παραγωγή του ενδιάμεσου κώδικα της μονάδας, γι' αυτό και η αρχική μορφή του πίνακα δέσμευσης αυτής δεν τις καλύπτει. Όπως και προηγουμένως, ο πίνακας δέσμευσης δεν παράγει εκτελέσιμο κώδικα, αλλά μόνο καθορίζει τις θέσεις και τα μεγέθη των δεδομένων που προσπελάζονται από τον κώδικα. Επειδή ο χώρος δεδομένων κάθε μονάδας εκτός της εξωτερικής αναφέρεται στη στοίβα, ο αντίστοιχος πίνακας δέσμευσης δεν είναι δυνατό να περιέχει αρχικές τιμές, και αυτές όπως προαναφέρθηκε αποδίδονται με εκφράσεις ανάθεσης.
Σύμφωνα με τα παραπάνω, ο πίνακας δέσμευσης της εξωτερικής μονάδας για το πρόγραμμα που μας δίνεται – με κάποιον αυθαίρετο συμβολισμό – θα είναι:

```
{ 1: int; 2: int; 3: int: 0 }
```

υποδηλώνοντας τρεις μεταβλητές τύπου int, οι οποίες αντιστοιχούν στις καθολικές μεταβλητές x και z, καθώς και στη στατική μεταβλητή x της συνάρτησης A. Η τελευταία από τις τρεις μεταβλητές έχει και αρχική τιμή, ίση με 0. Οι θέσεις των μεταβλητών είναι διαδοχικές, ξεκινώντας από τη θέση 1, και κάνοντας τη δέσμευση με τη σειρά που αυτές εμφανίζονται στο πρόγραμμα.
Παρατηρήστε ότι το όνομα των μεταβλητών δεν εμφανίζεται στον πίνακα, επειδή αυτό δε συμμετέχει στον τελικό κώδικα. Μέχρι να ολοκληρωθεί η μεταγλώττιση της μονάδας, όμως, και για να συνδέσουμε κάθε μεταβλητή με την αντίστοιχη δέσμευση, τοποθετούμε τον δείκτη στον πίνακα όπου αποθηκεύεται η παραπάνω πληροφορία και στον πίνακα συμβόλων. Για παράδειγμα, η εγγραφή στον πίνακα συμβόλων για την καθολική μεταβλητή x θα περιέχει την τιμή 1, εφ' όσον η πληροφορία δέσμευσης αυτής βρίσκεται στην πρώτη θέση του πίνακα δέσμευσης.
Ο πίνακας δέσμευσης της μονάδας της συνάρτησης A θα είναι:

```
{ 1: void*; 2: void; 3: int* }
```

δεσμεύοντας χώρο για τη διεύθυνση επανόδου (θέση 1), καθώς και για τη μοναδική παράμετρο (θέση 3). Η θέση 2 δεσμεύεται, ώστε να εξασφαλιστεί ομοιόμορφη αναφορά σε παραμέτρους για όλες τις συναρτήσεις της γλώσσας, άσχετα αν δε χρησιμοποιείται στην προκειμένη περίπτωση που η συνάρτηση A δεν επιστρέφει τιμή. Έτσι, η πρώτη παράμετρος θα βρίσκεται πάντα στη θέση 3, η δεύτερη στη θέση 4, κόκ. Παρατηρήστε ότι η τοπική μεταβλητή x δεν περιλαμβάνεται στον πίνακα, επειδή είναι στατική και έχει περιληφθεί στον πίνακα δέσμευσης της εξωτερικής μονάδας του προγράμματος.
Ο πίνακας δέσμευσης της μονάδας της συνάρτησης main θα είναι:

```
{ 1: void*; 2: void; 3: int; 4: int }
```

Οι δύο μεταβλητές που καλύπτονται από τον πίνακα αυτόν είναι η μεταβλητή y που δηλώνεται στην εμβέλεια της main (θέση 3), και η μεταβλητή j που δηλώνεται σε εμβέλεια εσωτερική της main (θέση 4). Η τελευταία μπορεί να έχει μικρότερη εμβέλεια από την y, αλλά ο χώρος που δεσμεύεται γι' αυτήν βρίσκεται στο εγγράφημα δραστηριοποίησης της main, κι επομένως η δέσμευση γίνεται στον ίδιο πίνακα δέσμευσης. Όπως και παραπάνω, έτσι και εδώ δεσμεύτηκαν δύο θέσεις για λόγους ομοιομορφίας, μια που στη γενική περίπτωση η main δέχεται παραμέτρους.

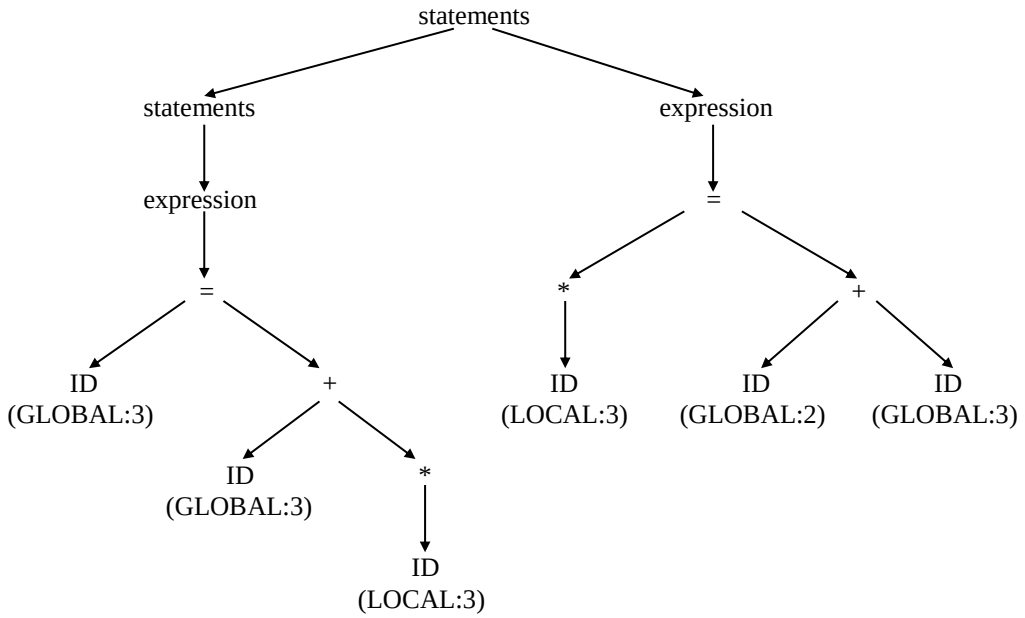
Οι δηλώσεις των μεταβλητών σε ένα πρόγραμμα C – και αφού προηγηθεί ο απαιτούμενος σημασιολογικός έλεγχος τύπων – μετατρέπονται από το μεταγλωττιστή σε δεσμεύσεις στο χώρο δεδομένων. Μια αναφορά σε δηλωμένη μεταβλητή, από την άλλη μεριά, θα αναπαρασταθεί στον ενδιάμεσο κώδικα από την ένδειξη του πίνακα όπου έχει δεσμευτεί χώρος γι' αυτή, και από τον αντίστοιχο δείκτη. Επειδή η C δεν υποστηρίζει φωλιάσματα συναρτήσεων, κάθε αναφορά σε μια μεταβλητή επιλύεται είτε στο τρέχον εγγράφημα δραστηριοποίησης στη στοίβα, είτε στο χώρο στατικών δεδομένων. Έτσι, η ένδειξη του πίνακα μπορεί να γίνει απλά με το χαρακτηρισμό GLOBAL, όταν η μεταβλητή τοποθετείται στο χώρο στατικών δεδομένων, ή από την ένδειξη LOCAL, όταν αυτή τοποθετείται στο εγγράφημα δραστηριοποίησης. Για παράδειγμα, στην έκφραση "x>y" της συνάρτησης main η αναφορά στην x αναπα-

ριστάται με το ζεύγος (GLOBAL: 1) ενώ η αναφορά στην y αναπαριστάται με το ζεύγος (LOCAL: 3). Σε κάθε αναφορά μιας μεταβλητής, η πληροφορία που καθορίζει τον πίνακα δέσμευσης, όπως και ο δείκτης σε αυτόν, βρίσκονται από τον πίνακα συμβόλων.
Στην παραγωγή του ενδιάμεσου κώδικα μπορούμε να υλοποιήσουμε μια διαδοχή από εντολές – ή αλλιώς μια σύνθετη εντολή – είτε σε μορφή δυαδικού δέντρου, χρησιμοποιώντας τους συνήθεις συντακτικούς κανόνες:

statements → statements statement | statement

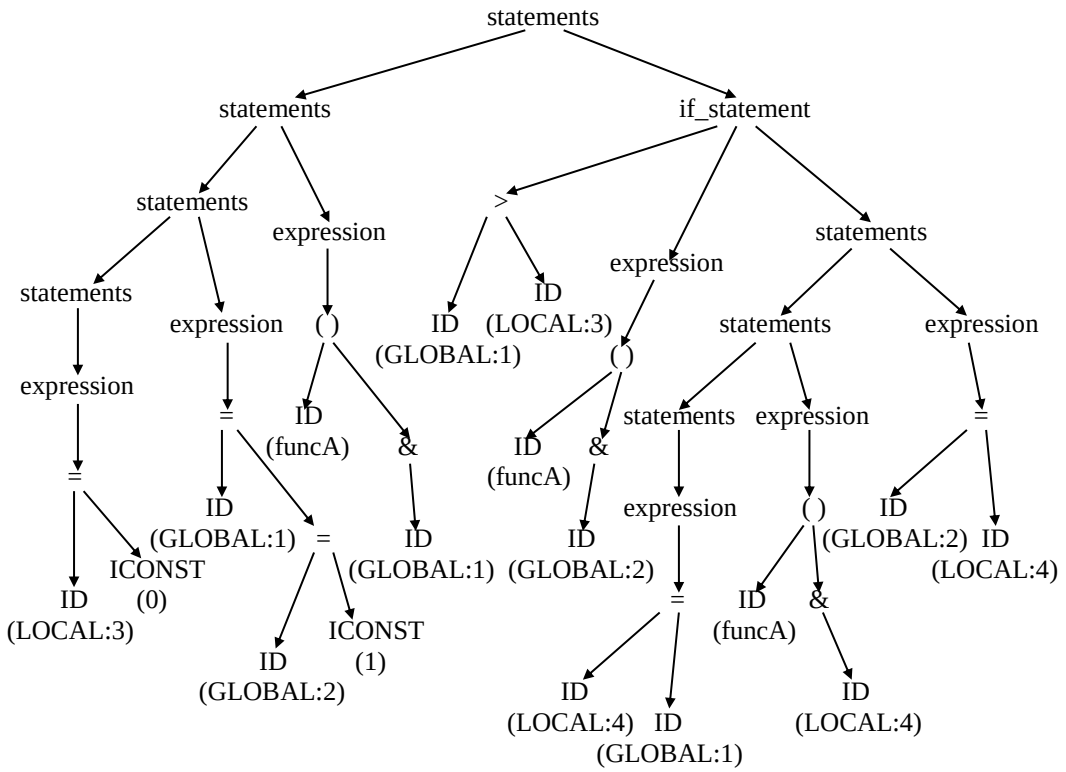
είτε σε μορφή γραμμικής λίστας. Θα προτιμήσουμε την πρώτη υλοποίηση, επειδή προσφέρει πιο γρήγορη προσπέλαση.
Κάθε δέντρο εντολών έχει στα φύλλα του εντολές του αρχικού προγράμματος. Ανάλογα με την εντολή, κάθε ένα φύλλο μπορεί να συνδέεται με ένα ή περισσότερα δέντρα εκφράσεων ή άλλα δέντρα εντολών, σχηματίζοντας έτσι ένα μεγαλύτερο δέντρο. Τα φύλλα του συνολικού δέντρου ενδιάμεσου κώδικα για μια μονάδα θα είναι τα φύλλα των δέντρων εκφράσεων της μονάδας, δηλαδή οι αναφορές σε αναγνωριστικά του προγράμματος, όπως για παράδειγμα μεταβλητές, καθώς και οι αναφορές σε σταθερές, οι οποίες απλά αναπαριστώνται με τις τιμές τους.
Τα δέντρα εκφράσεων, τέλος, κατασκευάζονται με βάση τους συντακτικούς κανόνες των εκφράσεων. Κάθε κόμβος δέντρου έκφρασης περιέχει έναν τελεστή και έχει σαν παιδιά του τις υποεκφράσεις που αποτελούν τα τελούμενα του τελεστή. Η ρίζα ενός δέντρου έκφρασης θα περιέχει έναν από τους πιθανά πολλούς τελεστές με τη μικρότερη προτεραιότητα στην έκφραση, ανάλογα με την προσεταιριστικότητα αυτού.

Έτσι, το αφηρημένο συντακτικό δέντρο που κατασκευάζεται σύμφωνα με τα παραπάνω για τη συνάρτηση A θα είναι το εξής:



όπου φαίνονται κωδικοποιημένες δύο εντολές έκφρασης, μια στα αριστερά που είναι η πρώτη εντολή της συνάρτησης A, και μία στα δεξιά που είναι η δεύτερη εντολή της A. Υπενθυμίζεται ότι στη C δεν υπάρχει ειδική εντολή ανάθεσης, και η ανάθεση αποτελεί μέρος εντολής έκφρασης. Παρατηρήστε ότι η αναφορά στη μεταβλητή x γίνεται για τη στατική μεταβλητή της A (δείκτης 3 στον πίνακα δέσμευσης της εξωτερικής μονάδας), και όχι για την καθολική μεταβλητή με το ίδιο όνομα (δείκτης 1 στον ίδιο πίνακα δέσμευσης).

Το αφηρημένο συντακτικό δέντρο που κατασκευάζεται για τη συνάρτηση main δίνεται στην επόμενη σελίδα. Σε αυτό φαίνονται κωδικοποιημένες τέσσερις εντολές, από τις οποίες οι τρεις πρώτες είναι εντολές έκφρασης, και η τέταρτη εντολή διακλάδωσης. Από τις εντολές



έκφρασης, η πρώτη περιέχει μια ανάθεση, η οποία υλοποιεί την αρχικοποίηση της μεταβλητής στοίβας y. Η δεύτερη περιέχει δύο αναθέσεις, που αντιστοιχούν στην εντολή:

```
x = z = 1;
```

της συνάρτησης main του αρχικού προγράμματος. Επειδή ο τελεστής ανάθεσης έχει δεξιά προσεταιριστικότητα, στη ρίζα του αντίστοιχου δέντρου έκφρασης βρίσκεται ο πρώτος από αριστερά από τους δύο τελεστές ανάθεσης. Η τρίτη εντολή έκφρασης είναι μια κλήση της συνάρτησης A. Η μετατροπή της εφαρμογής του τελεστή '( )' στα συγκεκριμένα βήματα με τα οποία πραγματοποιείται μια κλήση συνάρτησης γίνεται κατά την παραγωγή τελικού κώδικα. Στον ενδιάμεσο κώδικα το αναγνωριστικό μιας συνάρτησης σε μια κλήση προς αυτήν πρέπει να συνοδεύεται από την πληροφορία που παρέχει τουλάχιστον:

- Τις παραμέτρους της συνάρτησης, τον τύπο, καθώς και τον τρόπο περάσματος αυτών. Αυτή η πληροφορία μπορεί να δίνεται με ένα δείκτη στην εγγραφή της συνάρτησης στον πίνακα συμβόλων. Η εγγραφή θα είναι ορισμένη, αφού η εμβέλεια μιας συνάρτησης είναι όλο το πρόγραμμα[2], ακόμα κι αν η συνάρτηση είναι δηλωμένη μόνο σαν πρωτότυπο.
- Την αρχή του κώδικα της συνάρτησης. Αυτή η πληροφορία παρέχεται μέσω ενός δείκτη στον ενδιάμεσο κώδικα της μονάδας της συνάρτησης, και μπορεί επίσης να αποθηκεύεται στον πίνακα συμβόλων. Αν ο ενδιάμεσος κώδικας δεν έχει ακόμα παραχθεί, θα πρέπει να ξαναγυρίσουμε αργότερα στην κλήση αυτή με την τεχνική του μπαλώματος.

Μετά τις εντολές έκφρασης στον ενδιάμεσο κώδικα της main ακολουθεί μια εντολή διακλάδωσης. Αυτή αναπαριστάται με έναν κόμβο που έχει τρία παιδιά, μια έκφραση που αποτελεί τη συνθήκη άλματος της διακλάδωσης, και δύο εντολές, που μπορεί να είναι είτε απλές είτε σύνθετες. Στην προκειμένη περίπτωση η μία εντολή είναι απλή, και είναι μια εντολή έκφρασης, που περιέχει μια ακόμα κλήση της συνάρτησης A. Η δεύτερη εντολή είναι σύνθετη, κι επομένως υλοποιείται στον ενδιάμεσο κώδικα με ένα νέο δέντρο εντολών συνδεδεμένο με το προηγούμενο στον κόμβο της εντολής διακλάδωσης. Οι εντολές που απαρτίζουν τη σύνθετη αυτή εντολή είναι τρεις εντολές έκφρασης, από τις οποίες δύο περιέχουν αναθέσεις και μία περιέχει μια κλήση της συνάρτησης A.

[2] Για μείωση της πολυπλοκότητας της άσκησης δε θεωρούμε προγράμματα C που περιλαμβάνουν περισσότερα από ένα αρχεία, οπότε υπάρχουν εσωτερικές και εξωτερικές εμβέλειες για κάθε αρχείο.

Ενότητα 8: Βελτιστοποιήσεις

Άσκηση 8-1:

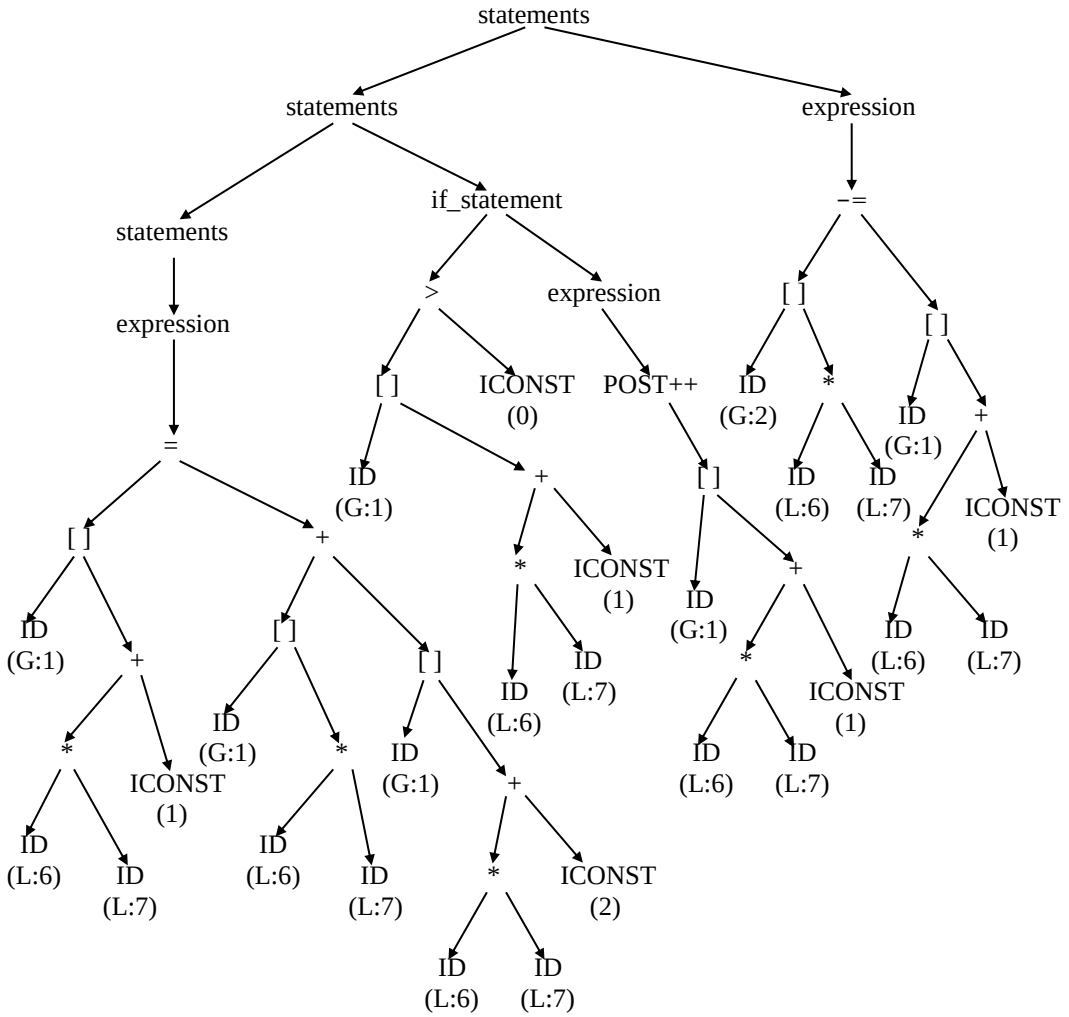
Έστω το πιο κάτω τμήμα κώδικα C:

```
a[i*j+1] = a[i*j] + a[i*j+2];
if (a[i*j+1] > 0) a[i*j+1]++;
b[i*j] -= a[i*j+1];
```

όπου a και b διανύσματα ακεραίων, και i και j ακέραιες μεταβλητές.
Α. Δώστε έναν ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου που να αντιστοιχεί στον κώδικα αυτόν, χωρίς την εφαρμογή οποιασδήποτε βελτιστοποίησης.
Β. Βρείτε τις κοινές υποεκφράσεις στον παραπάνω αρχικό κώδικα.
Γ. Δώστε ένα σχήμα εύρεσης και απαλοιφής των κοινών υποεκφράσεων στον ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου, λαμβάνοντας υπ' όψη τις διάρκειες ζωής των μεταβλητών που συμμετέχουν στον κώδικα.
Δ. Δείξτε την εφαρμογή του σχήματος αυτού στον πιο πάνω κώδικα.

Απάντηση

Α. Ο κώδικας που μας δίνεται περιλαμβάνει τρεις εντολές C. Η πρώτη και η τρίτη από αυτές είναι εντολές έκφρασης, ενώ η δεύτερη είναι εντολή διακλάδωσης.
Αν ένα δέντρο εντολών κωδικοποιείται σα δυαδικό δέντρο, ο ζητούμενος ενδιάμεσος κώδικας θα είναι:



Επειδή δεν έχουμε το συνολικό κώδικα, θεωρήσαμε αυθαίρετα – χωρίς να επηρεάζεται από αυτό η επίλυση της άσκησης – ότι οι αναφορές σε αναγνωριστικά μεταβλητών γίνονται στους χώρους δεδομένων που δείχνονται στον παραπάνω ενδιάμεσο κώδικα. Με G συμβολίσαμε το χώρο δεδομένων της εξωτερικής μονάδας του προγράμματος, ενώ με L συμβολίσαμε τον τοπικό χώρο δεδομένων, δηλαδή το εγγράφημα δραστηριοποίησης της παρούσας μονάδας του προγράμματος. Οι αριθμοί που συνοδεύουν τα σύμβολα αυτά αποτελούν δείκτες στους αντίστοιχους πίνακες δέσμευσης, του χώρου στατικών δεδομένων:

```
{ 1: int[1000]; 2: int[1000]; ... }
```

όπου θεωρήσαμε αυθαίρετα ως μέγεθος πινάκων τα 1000 στοιχεία, και του τοπικού χώρου δεδομένων:

```
{ 1: void*; 2: void; ... ; 6: int; 7: int; ... }
```

όπου δείχνονται – αν και δεν είναι απαραίτητο για την παρούσα άσκηση – και οι δύο πρώτες θέσεις για τη διεύθυνση επανόδου και την τιμή αποτελέσματος, αντίστοιχα.

Β. Για να εντοπίσουμε τις κοινές υποεκφράσεις στον κώδικα που μας δίνεται, πρέπει να βρούμε τις διάρκειες ζωής των μεταβλητών που συμμετέχουν σε αυτόν. Επειδή διαθέτουμε μόνο ένα απόσπασμα από το συνολικό κώδικα του προγράμματος, αρκεί να εξετάσουμε (α) αν οι μεταβλητές που συναντάμε δεν παίρνουν νέες τιμές στον κώδικα που διαθέτουμε, οπότε οι διάρκειες ζωής τους παρατείνονται τουλάχιστον μέχρι τα σημεία στον κώδικα όπου χρησιμοποιούνται για τελευταία φορά, ή (β) αν παίρνουν νέες τιμές σε αυτόν, οπότε και αλλάζουν διάρκεια ζωής στα σημεία στα οποία λαμβάνουν τις νέες τιμές.

Γενικά είναι εύκολο να αποφανθούμε αν μια βαθμωτή μεταβλητή αλλάζει τιμή σε κάποια εντολή, όταν αυτό *δεν* γίνεται μέσω μεταβλητής τύπου pointer. Στην προκειμένη περίπτωση, που δεν έχουμε μεταβλητές τύπου pointer στον κώδικά μας, βλέπουμε ότι οι μεταβλητές i και j δε λαμβάνουν τιμή σε καμιά από τις τρεις εντολές αυτού, κι επομένως η διάρκεια ζωής που έχουν μπαίνοντας στον κώδικα παρατείνεται τουλάχιστον μέχρι το τέλος του.
Για μη βαθμωτές μεταβλητές από την άλλη μεριά, και πιο συγκεκριμένα για μεταβλητές τύπου πίνακα, είναι ιδιαίτερα δύσκολο να αποφανθούμε αν συγκεκριμένα στοιχεία αυτών διατηρούν την τιμή τους μέσα σε κάποιον κώδικα, όταν οι μεταβλητές που συμμετέχουν στις εκφράσεις των δεικτών τους αλλάζουν τιμή μέσα σε αυτόν. Κάτι τέτοιο είναι μερικά εφικτό με τη χρήση ειδικών εργαλείων που αναλύουν συμβολικά τον κώδικα, εξετάζοντας τις τιμές που μπορούν να έχουν οι δείκτες των στοιχείων αυτών πριν και μετά από κάθε εντολή του κώδικα. Αν όμως οι εκφράσεις των δεικτών δεν αλλάζουν τιμή σε έναν κώδικα, είναι τότε εύκολο να αποφανθούμε αν τα στοιχεία που αναπαριστώνται με τους συγκεκριμένους αυτούς δείκτες αλλάζουν τιμή στον ίδιο κώδικα. Έτσι, για τον κώδικα που μας δίνεται, μπορούμε να διαπιστώσουμε ότι τα στοιχεία a[i*j] και a[i*j+2] του πίνακα a διατηρούν την τιμή τους και στις τρεις εντολές αυτού. Αντίθετα, το στοιχείο a[i*j+1] αλλάζει τιμή τόσο στην πρώτη, όσο και στη δεύτερη εντολή του κώδικα. Ειδικά για τη δεύτερη εντολή, μας αρκεί το γεγονός ότι για κάποια τιμή της συνθήκης άλματος είναι δυνατό το στοιχείο αυτό να αλλάξει τιμή, ώστε να πούμε ότι το στοιχείο αυτό *γενικά* αλλάζει τιμή στην εντολή αυτή. Το στοιχείο b[i*j] του πίνακα b τέλος, αλλάζει τιμή στην τρίτη εντολή του κώδικα.

Οι εκφράσεις ενός κώδικα χωρίζονται σε δύο κατηγορίες: εκφράσεις αριστερής και εκφράσεις δεξιάς προσπέλασης. Στην πρώτη κατηγορία ανήκουν εκφράσεις στις οποίες μπορεί να γίνει ανάθεση κάποιας τιμής, όπως είναι για παράδειγμα εκφράσεις διευθύνσεων μεταβλητών, είτε βαθμωτών είτε στοιχείων πίνακα, ενώ στη δεύτερη κατηγορία ανήκουν οι υπόλοιπες.
Ο διαχωρισμός στις δύο κατηγορίες είναι απαραίτητος για την εύρεση κοινών υποεκφράσεων, επειδή οι διάρκειες ζωής των μεταβλητών χρησιμοποιούνται διαφορετικά σε καθεμία από αυτές. Πιο συγκεκριμένα, αν σε έναν κώδικα δε συμμετέχουν μεταβλητές τύπου pointer, οι τιμές αριστερής προσπέλασης δεν επηρεάζονται από αναθέσεις στις αντίστοιχες μεταβλητές, κι έτσι η διάρκεια ζωής τους είτε δεν αλλάζει, αν οι μεταβλητές είναι βαθμωτές, είτε καθορί-

ζεται από τις διάρκειες ζωής των μεταβλητών που συμμετέχουν στις εκφράσεις των δεικτών, αν οι μεταβλητές είναι στοιχεία πίνακα. Η διάρκεια ζωής τιμών δεξιάς προσπέλασης, από την άλλη μεριά, λαμβάνει υπ' όψη τις αναθέσεις σε όλες τις μεταβλητές που συμμετέχουν στις εκφράσεις τους, και αλλάζει με κάθε ανάθεση σε μια από αυτές.
Οι μεταβλητές τύπου pointer καθιστούν την παραπάνω ανάλυση πιο δύσκολη, επειδή οι τιμές τους είναι ταυτόχρονα αριστερής και δεξιάς προσπέλασης, και ανάθεση σε μια τέτοια μεταβλητή αλλάζει τη διεύθυνση στην οποία αυτή δείχνει. Επειδή δε η τιμή που ανατίθεται είναι γενικά άγνωστη κατά τη μετάφραση, κάθε επόμενη ανάθεση στη διεύθυνση που δείχνει η μεταβλητή τύπου pointer πρέπει να τερματίζει όλες τις ενεργές διάρκειες ζωής που πιθανά επηρεάζονται.

Στον κώδικα που μας δίνεται, οι εκφράσεις αριστερής προσπέλασης είναι: (α) οι διευθύνσεις των βαθμωτών μεταβλητών i και j, και (β) οι διευθύνσεις των στοιχείων πίνακα a[i*j], a[i*j+1], a[i*j+2] και b[i*j]. Οι διευθύνσεις των δύο βαθμωτών μεταβλητών που εμφανίζονται σε περισσότερα από ένα σημεία στον κώδικα, αποτελούν κοινές υποεκφράσεις σε αυτόν, επειδή είναι σταθερές, κι επομένως οι διάρκειες ζωής τους δεν αλλάζουν. Εφ' όσον οι μεταβλητές i και j δεν παίρνουν τιμή σε καμιά εντολή του, όλες οι εκφράσεις των δεικτών έχουν διάρκειες ζωής που επίσης δεν αλλάζουν στον κώδικα, κι επομένως όλες οι εκφράσεις αριστερής προσπέλασης στοιχείων πίνακα που εμφανίζονται περισσότερες της μίας φορές αποτελούν κοινές υποεκφράσεις σε αυτόν. Αυτό συμβαίνει μόνο με το στοιχείο a[i*j+1], η διεύθυνση του οποίου αποτελεί επομένως κοινή υποέκφραση στον κώδικα.
Όσο αφορά τώρα τις εκφράσεις δεξιάς προσπέλασης, κοινές υποεκφράσεις αποτελούν (α) όλες οι σταθερές που χρησιμοποιούνται περισσότερες από μία φορές στον κώδικα, (β) όλες οι τιμές μεταβλητών που χρησιμοποιούνται περισσότερες από μία φορές στην ίδια διάρκεια ζωής στον κώδικα, και (γ) όλες οι εκφράσεις που κατασκευάζονται από τις προηγούμενες και, όπως οι μεταβλητές, χρησιμοποιούνται περισσότερες από μία φορές στην ίδια διάρκεια ζωής. Έτσι, η σταθερά 1, οι τιμές των μεταβλητών i και j, και οι εκφράσεις "i*j" και "i*j+1" αποτελούν κοινές υποεκφράσεις σε όλον τον κώδικα που μας δίνεται. Επιπλέον, η τιμή του στοιχείου πίνακα a[i*j+1] αποτελεί κοινή υποέκφραση στη δεύτερη εντολή του κώδικα, η οποία τη χρησιμοποιεί δύο φορές στη διάρκεια ζωής που ξεκινάει με την ανάθεση κάποιας τιμής στο στοιχείο αυτό στην πρώτη εντολή και τελειώνει με τη χρήση αυτής της τιμής στο τέλος της δεύτερης εντολής του κώδικα.

Γ. Για την εύρεση των κοινών υποεκφράσεων στον ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου, είναι απαραίτητο να έχει προηγηθεί ανάλυση ροής ελέγχου και ροής δεδομένων για τον υπολογισμό της διάρκειας ζωής των μεταβλητών του κώδικα. Υποθέτουμε ότι η ανάλυση αυτή καλύπτει μεταβλητές τύπου πίνακα, χωρίς όμως να συμπεριλαμβάνει συμβολική ανάλυση των τιμών των δεικτών τους.

Με την υπόθεση ότι η μετάφραση που χρησιμοποιείται είναι οδηγούμενη από τη σύνταξη, και ότι η βάση της ανάλυσης είναι ένας συντακτικός αναλυτής από κάτω προς τα επάνω, το αφηρημένο συντακτικό δέντρο κατασκευάζεται από κάτω προς τα επάνω, και από αριστερά προς τα δεξιά. Έτσι, οι κοινές υποεκφράσεις μπορούν να βρεθούν κατά την κατασκευή των δέντρων εκφράσεων με τον αλγόριθμο που δίνεται παρακάτω:

Κατασκεύασε το κόμβο έκφρασης τύπου Τ με παιδιά Π1 και Π2 – αν υπάρχουν, ως εξής:
- Αν Τ είναι τύπος σταθεράς, ψάξε το δέντρο για ένα φύλλο σταθεράς με την ίδια τιμή. Αν υπάρχει τέτοιο φύλλο, επέστρεψε αυτό το φύλλο, αλλιώς κατασκεύασε και επέστρεψε ένα νέο φύλλο σταθεράς.
- Αν Τ είναι τύπος αναγνωριστικού, ψάξε το δέντρο για ένα φύλλο αναγνωριστικού της ίδιας μεταβλητής. Αν η έκφραση είναι δεξιάς προσπέλασης, το φύλλο πρέπει να βρίσκεται στην ίδια διάρκεια ζωής της μεταβλητής με αυτή, στην οποία τώρα βρισκόμαστε. Αν υπάρχει τέτοιο φύλλο, επέστρεψε αυτό το φύλλο, αλλιώς κατασκεύασε και επέστρεψε ένα νέο φύλλο αναγνωριστικού.

- Αν κάποιο από τα παιδιά Π1 και Π2 δεν έχει πατέρα, ή αν τα δύο παιδιά δεν έχουν τον ίδιο πατέρα τύπου Τ, κατασκεύασε και επέστρεψε ένα νέο κόμβο έκφρασης τύπου Τ με παιδιά Π1 και Π2.
- Αν Τ είναι τύπος τελεστή που αποτιμά έκφραση αριστερής προσπέλασης σε δεξιάς, ψάξε για έναν κοινό πατέρα στην ίδια διάρκεια ζωής της μεταβλητής που αντιστοιχεί στην έκφραση αριστερής προσπέλασης, με αυτή, στην οποία τώρα βρισκόμαστε. Αν υπάρχει τέτοιος κόμβος, επέστρεψε αυτόν τον κόμβο. Διαφορετικά, αν υπάρχει κοινός πατέρας έκφρασης αριστερής προσπέλασης, επέστρεψε αυτόν τον κόμβο, αλλιώς κατασκεύασε και επέστρεψε ένα νέο κόμβο έκφρασης τύπου Τ με παιδιά Π1 και Π2.
- Επέστρεψε τον κοινό πατέρα των Π1 και Π2.

Ο πιο πάνω αλγόριθμος, όταν προσθέτει στον ενδιάμεσο κώδικα έναν κόμβο που αποτιμά μια υποέκφραση, διαπερνά το δέντρο που έχει ήδη κατασκευαστεί, για να βρει έναν κόμβο που αποτιμά μια κοινή με αυτήν υποέκφραση. Οι δύο πρώτοι έλεγχοι εξασφαλίζουν ότι αν ο κόμβος αντιστοιχεί σε φύλλο του δέντρου, αυτός θα προστεθεί μόνο αν δεν υπάρχει άλλο ίδιο φύλλο που να έχει προστεθεί νωρίτερα. Με τους επόμενους έλεγχους, κι επειδή η κατασκευή γίνεται από κάτω προς τα επάνω, ο αλγόριθμος εξασφαλίζει ότι τα παιδιά του κόμβου δεν προέρχονται από κοινές με αυτά υποεκφράσεις με τον ίδιο πατέρα, πριν προσθέσει κάποιο νέο κόμβο. Έτσι, όταν τελικά κατασκευάζει ένα νέο κόμβο, είναι σίγουρος ότι δεν υπάρχει άλλος κόμβος στον ενδιάμεσο κώδικα που αποτιμά την ίδια υποέκφραση. Ως αποτέλεσμα των παραπάνω, η τελευταία επιλογή στον αλγόριθμο δεν είναι δυνατό να έχει να επιλέξει περισσότερους από έναν κοινούς πατέρες των Π1 και Π2.
Ειδικά για τις υποεκφράσεις δεξιάς προσπέλασης, ο αλγόριθμος λαμβάνει υπ' όψη τις διάρκειες ζωής των μεταβλητών που συμμετέχουν σε αυτές, βρίσκοντας έτσι μόνο εκείνες τις κοινές υποεκφράσεις, οι οποίες αντιστοιχούν σε κοινές διάρκειες ζωής όλων των μεταβλητών. Αν δεν υπάρχει προηγούμενη ίδια υποέκφραση δεξιάς προσπέλασης, ο αλγόριθμος εξετάζει την ύπαρξη ίδιας υποέκφρασης αριστερής προσπέλασης. Οι κοινές υποεκφράσεις που παράγονται με αυτόν τον τρόπο διαφοροποιούνται στην παραγωγή τελικού κώδικα, είτε με διαφορετική επιλογή εντολών είτε με την προσθήκη κατάλληλου κόμβου για μετατροπή της έκφρασης αριστερής σε έκφραση δεξιάς προσπέλασης, και κάτι τέτοιο πρέπει να σημειώνεται στον ενδιάμεσο κώδικα με κατάλληλη ένδειξη.
Ας σημειωθεί ότι το αφηρημένο συντακτικό δέντρο που κατασκευάζεται με τον παραπάνω τρόπο δεν έχει ακριβώς δομή δέντρου, αλλά κατευθυνόμενου γράφου, αφού ένας κόμβος μπορεί να έχει περισσότερους από έναν πατέρες. Έτσι, απαλείφονται οι κοινές υποεκφράσεις από τον κώδικα, αφού κάθε μία αποτιμάται μόνο μια φορά σε αυτόν.
Ακόμα, θα πρέπει να αναφέρουμε ότι ανάλογα με την τελική γλώσσα μεταγλώττισης, το πρώτο βήμα του αλγόριθμου μπορεί να παραλειφθεί. Για παράδειγμα, στην αρχιτεκτονική MIPS πολλές σταθερές ενσωματώνονται σε εντολές αριθμητικών και λογικών πράξεων ως άμεσα τελούμενα, κάτι που στην ουσία σημαίνει ότι οι κόμβοι τέτοιων σταθερών στον ενδιάμεσο κώδικα δε θα παράγουν ξεχωριστή εντολή στον τελικό κώδικα, και άρα δε χρειάζεται να συμμετάσχουν στη συγκεκριμένη βελτιστοποίηση.

Στο ίδιο αποτέλεσμα καταλήγουμε για εύρεση κοινών υποεκφράσεων σε ήδη κατασκευασμένο αφηρημένο συντακτικό δέντρο, προχωρώντας με διαπέραση αυτού κατά βάθος και από αριστερά προς τα δεξιά. Τα βήματα του αλγόριθμου είναι παρόμοια, με τις ακόλουθες διαφορές:

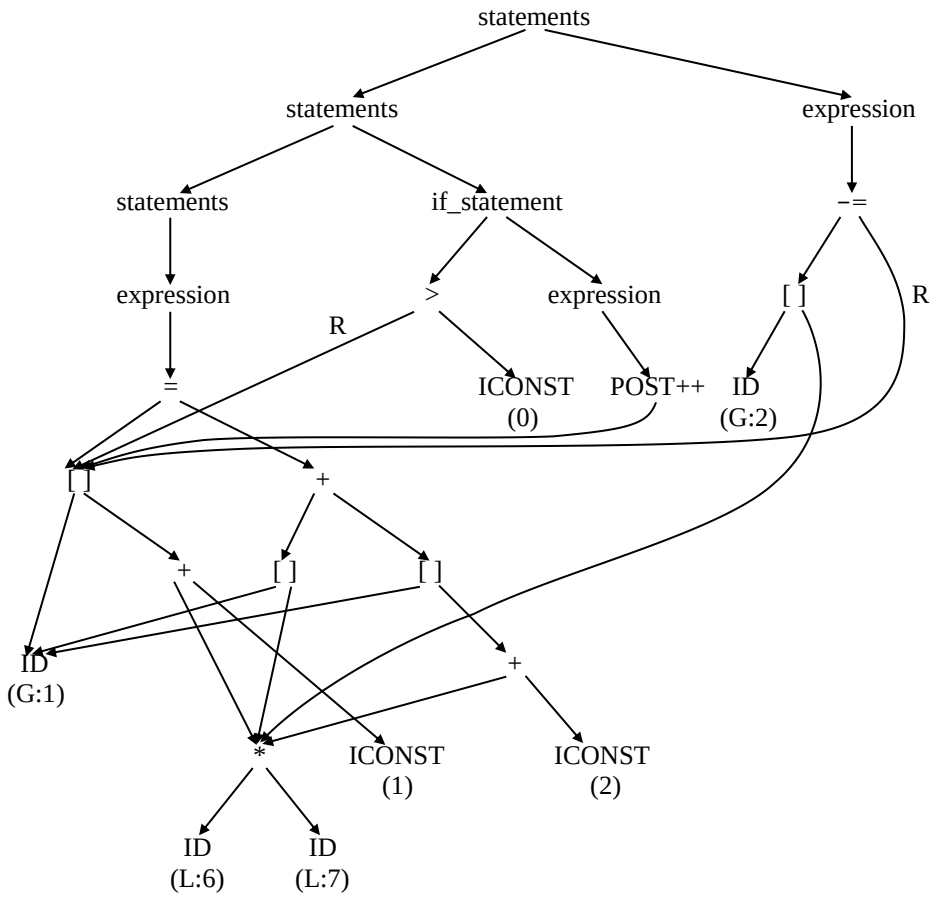
- Η όποια αναζήτηση κοινών υποεκφράσεων γίνεται στο υποδέντρο που έχουμε ήδη επισκεφτεί.
- Η επιστροφή – ως αποτέλεσμα της αναζήτησης – ενός κόμβου που έχουμε νωρίτερα επισκεφτεί οδηγεί στη διαγραφή του κόμβου στον οποίο τώρα βρισκόμαστε και στην αντικατάσταση των συνδέσεων προς τον τελευταίο με συνδέσεις προς τον κόμβο που επιστράφηκε.
- Δεν κατασκευάζεται κανένας νέος κόμβος.

Δ. Για την εξ αρχής κατασκευή του αφηρημένου συντακτικού δέντρου για το δεδομένο κώδικα με βάση τον πιο πάνω αλγόριθμο, ακολουθούμε τα εξής βήματα:

Βήμα	Περιγραφή
1	Κατασκευή κόμβου ID(a).
2	Κατασκευή κόμβου ID(i).
3	Κατασκευή κόμβου ID(j).
4	Κατασκευή κόμβου πολλαπλασιασμού.
5	Κατασκευή κόμβου ICONST(1).
6	Κατασκευή κόμβου πρόσθεσης.
7	Κατασκευή κόμβου υπολογισμού διεύθυνσης στοιχείου πίνακα a[i*j+1].
8-11	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 1-4.
12	Κατασκευή κόμβου αποτίμησης στοιχείου πίνακα a[i*j].
13-16	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 1-4.
17	Κατασκευή κόμβου ICONST(2).
18	Κατασκευή κόμβου πρόσθεσης.
19	Κατασκευή κόμβου αποτίμησης στοιχείου πίνακα a[i*j+2].
20	Κατασκευή κόμβου πρόσθεσης.
21	Κατασκευή κόμβου ανάθεσης.
22	Κατασκευή κόμβου εντολής έκφρασης.
23	Κατασκευή κόμβου σύνθετης εντολής.
24-29	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 1-6.
30	Επιστροφή κόμβου που κατασκευάστηκε στο βήμα 7 με ένδειξη ‘R’.
31	Κατασκευή κόμβου ICONST(0).
32	Κατασκευή κόμβου σύγκρισης.
33-39	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 1-7.
40	Κατασκευή κόμβου αύξησης.
41	Κατασκευή κόμβου εντολής έκφρασης.
42	Κατασκευή κόμβου εντολής διακλάδωσης.
43	Κατασκευή κόμβου σύνθετης εντολής.
44	Κατασκευή κόμβου ID(b).
45-47	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 2-4.
48	Κατασκευή κόμβου υπολογισμού διεύθυνσης στοιχείου πίνακα b[i*j].
49-54	Επιστροφή κόμβων που κατασκευάστηκαν στα βήματα 1-6.
55	Επιστροφή κόμβου που κατασκευάστηκε στο βήμα 7 με ένδειξη ‘R’.
56	Κατασκευή κόμβου αφαίρεσης και ανάθεσης.
57	Κατασκευή κόμβου εντολής έκφρασης.
58	Κατασκευή κόμβου σύνθετης εντολής.

όπου η ένδειξη ‘R’ υποδεικνύει την ανάγκη μετατροπής κοινής υποέκφρασης αριστερής προσπέλασης σε δεξιά.
Παρατηρήστε ότι με τα παραπάνω βήματα κατασκευάζονται μόνο 26 κόμβοι ενδιάμεσου κώδικα από τους 58 που κατασκευάστηκαν αρχικά. Αυτό φαίνεται στο διάγραμμα του νέου ενδιάμεσου κώδικα που ακολουθεί στην επόμενη σελίδα, όχι πια με μορφή δέντρου, αλλά κατευθυνόμενου γράφου.

Αξίζει να σημειώσουμε ότι στον κώδικά μας το μοναδικό παιδί του κόμβου ‘++’ και το αριστερό παιδί του κόμβου ‘−=’ είναι κόμβοι εκφράσεων τόσο δεξιάς όσο και αριστερής προσπέλασης, μια που ο κώδικας χρειάζεται και τις τιμές – για την αριθμητική πράξη, αλλά και τις διευθύνσεις – για την αποθήκευση – των αντίστοιχων μεταβλητών. Θα μπορούσαμε να είχαμε διαχωρίσει τις δεξιές από τις αριστερές προσπελάσεις με κατάλληλους μετασχηματισμούς σε κώδικα με διαφορετικό τον τελεστή της πράξης από τον τελεστή της ανάθεσης, αλλά κάτι τέτοιο δεν το κάναμε, επειδή εξαρτάται σε μεγάλο βαθμό από την τελική αρχιτεκτο-



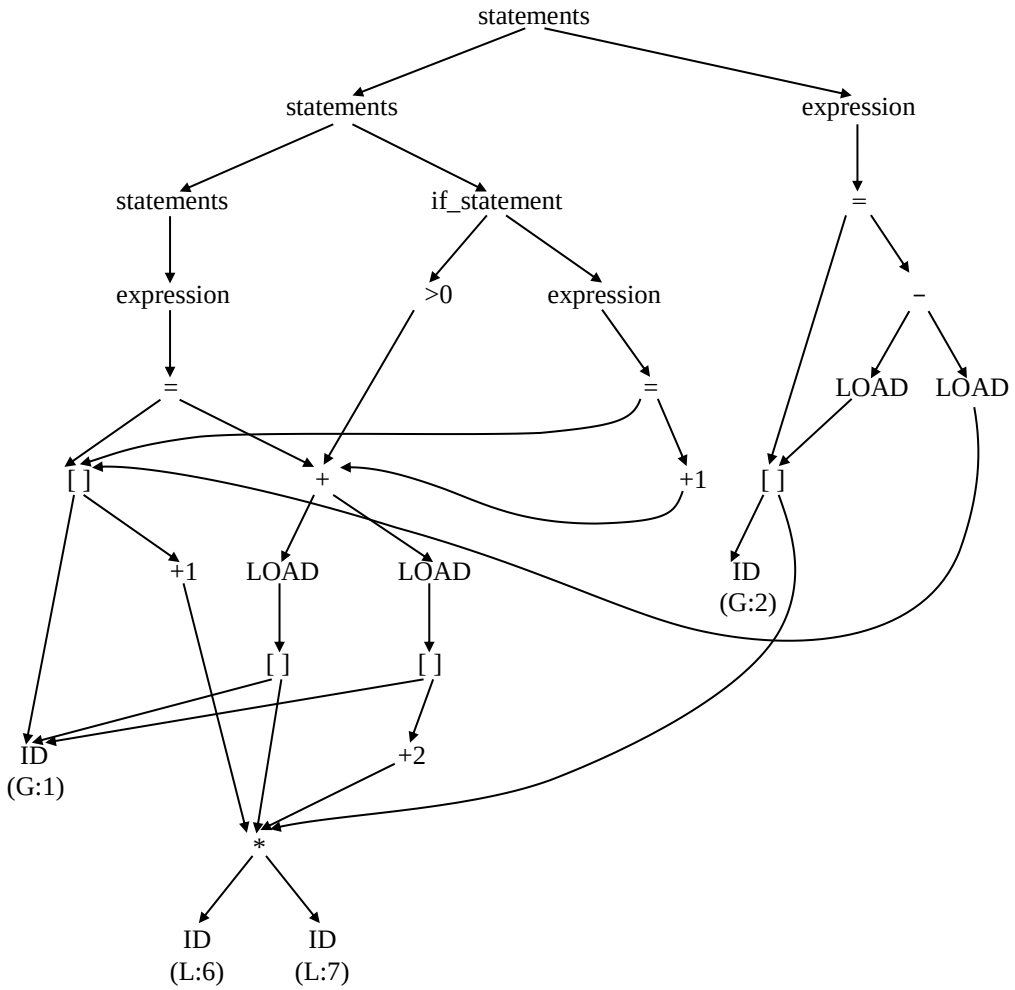
νική για την οποία γίνεται η μεταγλώττιση, και κάποιες φορές ο διαχωρισμός αυτός δεν είναι επιθυμητός[3]. Αν όμως είχαμε κάνει αυτό το διαχωρισμό, θα είχαμε δει και την τιμή του στοιχείου πίνακα a[i*j+1] σαν κοινή υποέκφραση, όπως αναφέραμε στο μέρος Β της άσκησης, και όχι μόνο τη διεύθυνσή του.

Αν όμως γνωρίζουμε στη φάση αυτή ότι η τελική αρχιτεκτονική είναι τύπου RISC, μπορούμε να κάνουμε το διαχωρισμό των δύο τύπων προσπέλασης ξεκάθαρο, με εισαγωγή ειδικού τελεστή φόρτωσης από τη μνήμη, οπουδήποτε λαμβάνεται τιμή δεξιάς προσπέλασης από έκφραση αριστερής προσπέλασης.
Θα μπορούσαμε τέλος να εφαρμόσουμε και διάδοση αντιγράφων, ώστε μια τέτοια τιμή δεξιάς προσπέλασης να λαμβάνεται από την αρχή της αντίστοιχης διάρκειας ζωής, απ' ευθείας από το δεξί μέλος της ανάθεσης που την ορίζει, αν βέβαια η ανάθεση αυτή είναι μοναδική, γλιτώνοντας έτσι το κόστος μιας φόρτωσης από τη μνήμη, που είναι μάλιστα εξαρτημένη από προηγούμενη αποθήκευση. Στον κώδικά μας, διάδοση αντιγράφων εφαρμόζεται στην τιμή του στοιχείου a[i*j+1] που χρησιμοποιούν οι τελεστές '>' και '++', αλλά όχι σε αυτή που χρησιμοποιεί ο τελεστής '−=', μια που στην περίπτωση αυτή η τιμή δεξιάς προσπέλασης μπορεί να ληφθεί από δύο διαφορετικές αναθέσεις.

Σε περίπτωση λοιπόν που γνωρίζουμε ότι η μεταγλώττιση γίνεται για κάποια αρχιτεκτονική RISC, μπορούμε να καταλήξουμε στον ενδιάμεσο κώδικα που ακολουθεί, όπου με LOAD παριστάνεται ο τελεστής φόρτωσης. Εκτός των άλλων, η αύξηση κατά σταθερά και η σύ-

[3] Σε αρχιτεκτονικές RISC για παράδειγμα, οι τελεστές '++' και '−=' αποτιμώνται με φόρτωση, πρόσθεση/αφαίρεση σταθερών ή καταχωρητών και αποθήκευση. Η παραγωγή τελικού κώδικα απαιτεί έτσι τρεις λειτουργίες, οι οποίες μπορούν να προκύψουν πιο εύκολα από τρεις κόμβους ενδιάμεσου κώδικα παρά από έναν. Οι αρχιτεκτονικές CISC από την άλλη μεριά, εκτελούν λειτουργίες απ' ευθείας στη μνήμη, οπότε αρκούνται στον ένα κόμβο, και δε χρειάζονται μετατροπή του κώδικα.

γκριση με το 0 έχουν ενσωματωθεί στον πατρικό κόμβο πρόσθεσης και σύγκρισης αντίστοιχα, αφού τέτοιες πράξεις γίνονται συνήθως μέσω εντολών με άμεσο τελούμενο.



Ενότητα 9: Τελικός κώδικας

Άσκηση 9-1:

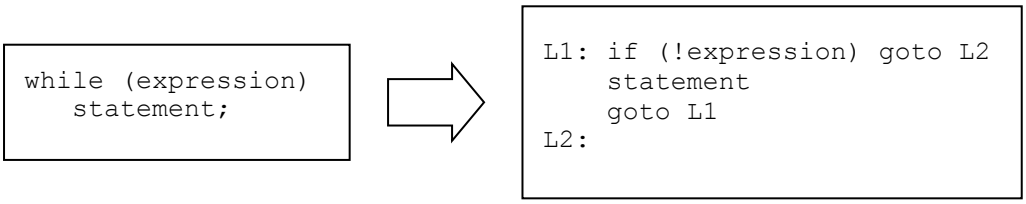
Να δώσετε ένα σχήμα παραγωγής τελικού κώδικα MIPS για την εντολή βρόχου while της C, της οποίας η σύνταξη δίνεται από τον κανόνα:

```
while → T_WHILE '(' expression ')' statement
```

όπου τα σύμβολα expression και statement περιγράφουν τις εκφράσεις και τις εντολές της C, αντίστοιχα. Υποθέστε ότι διαθέτετε ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου.

Απάντηση

Ένα σχήμα παραγωγής τελικού κώδικα περιγράφεται από τα βήματα που πρέπει να εκτελέσει ο μεταγλωττιστής για να μετατρέψει έναν ενδιάμεσο κώδικα σε τελικό.
Γενικά, για ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου, τα βήματα αυτά περιλαμβάνουν μια σειρά από μετασχηματισμούς που "κατεβάζουν" το επίπεδο των κόμβων του δέντρου, ώστε από κάποιο υψηλό επίπεδο, κοντά δηλαδή στην αρχική γλώσσα, να φτάσουν σε ένα επίπεδο πολύ κοντά στην τελική γλώσσα.
Για παράδειγμα, η εντολή while περιγράφεται στον ενδιάμεσο κώδικα από έναν κόμβο τύπου *εντολής while* και δύο παιδιά, ένα δείκτη σε ένα δέντρο έκφρασης, κι ένα δείκτη σε μια εντολή ή ένα δέντρο εντολών. Ο τελικός όμως κώδικας δεν έχει καμία εντολή που να υλοποιεί απ' ευθείας την εντολή while, κι έτσι η εντολή αυτή υλοποιείται τελικά με τη βοήθεια εντολών διακλάδωσης και άμεσου άλματος:

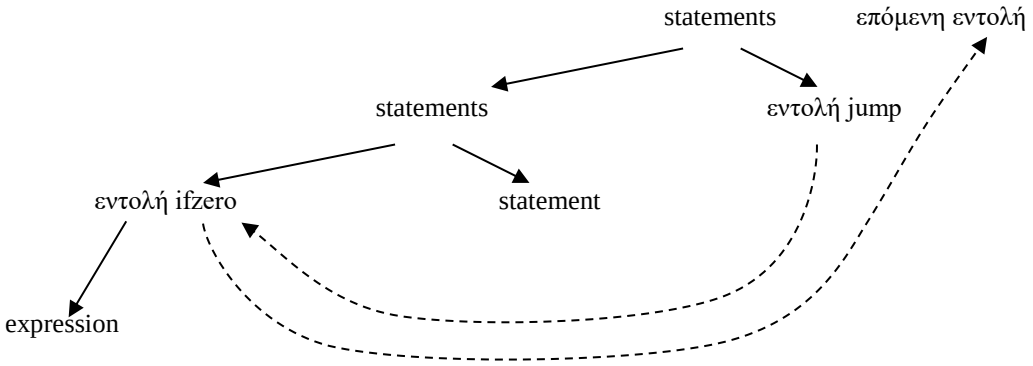


Για την εύρεση του ζητούμενου σχήματος δε μας απασχολεί η μετάφραση των παιδιών του κόμβου της εντολής, κάτι που γίνεται με εφαρμογή άλλων σχημάτων παραγωγής τελικού κώδικα.

Το σχήμα παραγωγής τελικού κώδικα της εντολής while βασίζεται στο μετασχηματισμό που δείχνει το παραπάνω διάγραμμα και περιλαμβάνει κατ' αρχήν τα ακόλουθα βήματα:

1. Κατασκευή ενός νέου κόμβου κάποιας απλής εντολής που εκτελεί άλμα αν το τελούμενό της έχει μηδενική τιμή. Τέτοια απλή εντολή διατίθεται στην αρχιτεκτονική MIPS, αλλά και στις περισσότερες σύγχρονες αρχιτεκτονικές. Έστω ότι ονομάζουμε τον τύπο του κόμβου αυτού *εντολή ifzero* (ή αλλιώς *εντολή iffalse*). Ο κόμβος αυτός θα έχει δύο παιδιά, ένα δείκτη στο δέντρο έκφρασης της αρχικής εντολής while, και ένα δείκτη στον κόμβο της εντολής που ακολουθεί την εντολή while.
2. Σύνδεση σε δυαδικό δέντρο του νέου κόμβου, με τον κόμβο εντολής statement που αντιστοιχεί στην εντολή ή στο δέντρο εντολών της εντολής while.
3. Κατασκευή ενός νέου κόμβου εντολής άμεσου άλματος, με τύπο που ονομάζουμε *εντολή jump*. Ο κόμβος θα έχει ένα παιδί, ένα δείκτη στον πρώτο κόμβο που κατασκευάσαμε.
4. Σύνδεση σε δυαδικό δέντρο του προηγούμενου δέντρου των δύο κόμβων με το νέο κόμβο.
5. Αντικατάσταση του αρχικού κόμβου της εντολής while με το δέντρο των τριών κόμβων που κατασκευάσαμε.

Μετά την εφαρμογή του πιο πάνω σχήματος, ο νέος κώδικας που λαμβάνουμε έχει την παρακάτω μορφή:



Από τη μορφή αυτή συνεχίζουμε με τις όποιες βελτιστοποιήσεις, και προχωράμε στην παραγωγή τελικού κώδικα με δέσμευση καταχωρητών.
Αν στον ενδιάμεσο κώδικα έχουν γίνει βελτιστοποιήσεις που εντοπίζουν κοινές υποεκφράσεις, είναι αναγκαία μια δέσμευση καταχωρητών στο συνολικό δέντρο του κώδικα. Μια τέτοια δέσμευση – για την περίπτωση βρόχου που έχουμε εδώ – απαιτεί πολύπλοκη ανάλυση ροής ελέγχου και δεδομένων και δε θα μας απασχολήσει στην παρούσα άσκηση. Έτσι, θα υποθέσουμε ότι η δέσμευση καταχωρητών για την έκφραση expression και την εντολή statement γίνεται ανεξάρτητα για τα δύο υποδέντρα, από τα αντίστοιχα σχήματα παραγωγής τελικού κώδικα.
Δεδομένου ότι μετά την αποτίμηση της έκφρασης expression η τιμή της θα περιέχεται σε κάποιον καταχωρητή της αρχιτεκτονικής, η πρώτη εντολή τελικού κώδικα που θα εκτελεστεί στη συνέχεια θα παραχθεί από τον πρώτο κόμβο του νέου δέντρου που κατασκευάσαμε, και θα έχει σαν τελούμενο εισόδου τον καταχωρητή αυτόν. Ο καταχωρητής που περιέχει την τιμή της έκφρασης expression είναι και ο μοναδικός καταχωρητής που χρησιμοποιούμε για την εκτέλεση της εντολής while, εκτός βέβαια από όσους χρησιμοποιούμε για την αποτίμηση της έκφρασης, και από όσους χρησιμοποιεί η εντολή statement. Η εντολή άμεσου άλματος δε χρησιμοποιεί καταχωρητές.

Σαν τελευταίο βήμα, ο τελικός κώδικας λαμβάνεται με διαπέραση του παραπάνω δέντρου κατά βάθος, και από αριστερά προς τα δεξιά. Ο αλγόριθμος διαπέρασης αγνοεί τις συνδέσεις που αντιστοιχούν σε άλματα, γι' αυτό και αυτές έχουν παρασταθεί παραπάνω με διάστικτες γραμμές. Η επιλογή των εντολών MIPS για κάθε κόμβο του δέντρου γίνεται στο βήμα αυτό.
Όσο αφορά τους ακριβείς προορισμούς των αλμάτων στους δύο νέους κόμβους που εισάγαμε, αυτοί θα είναι (α) για τον πρώτο, η πρώτη εντολή που εκτελείται στο κόμβο της εντολής που ακολουθεί την εντολή while, και (β) για το δεύτερο, η πρώτη εντολή που εκτελείται για την αποτίμηση της έκφρασης expression. Οι προορισμοί αυτοί δίνονται με ετικέτες, αν ο τελικός κώδικας παράγεται σε συμβολική μορφή. Διαφορετικά, κωδικοποιούνται στις αντίστοιχες εντολές, και μάλιστα με χρήση της τεχνικής του μπαλώματος για τον πρώτο κόμβο, αφού τη στιγμή επεξεργασίας του δε γνωρίζουμε πόσες εντολές μεσολαβούν μέχρι τον προορισμό του άλματος που περιέχει.

Έτσι, ένα απόσπασμα εντολών MIPS που μπορούμε να λάβουμε από το πιο πάνω σχήμα παραγωγής τελικού κώδικα είναι το παρακάτω:

```
L1: <κώδικας αποτίμησης έκφρασης, με αποτέλεσμα στον καταχωρητή $8>
    beq $8,$0,L2
    <κώδικας εντολής statement>
    j L1
L2: <κώδικας εντολής που ακολουθεί την εντολή while>
```

Άσκηση 9-2:

Να δώσετε ένα σχήμα παραγωγής τελικού κώδικα MIPS για την αποτίμηση των λογικών εκφράσεων της γλώσσας C με βραχυκύκλωση.
Υποθέστε ότι ο ενδιάμεσος κώδικας έχει τη μορφή αφηρημένου συντακτικού δέντρου, όπου οι εκφράσεις αναπαριστώνται με δέντρα εκφράσεων.

Απάντηση

Η ιδιομορφία της βραχυκύκλωσης στην αποτίμηση των λογικών εκφράσεων της C, αλλά και πολλών άλλων γλωσσών προγραμματισμού, είναι ότι διακόπτει την αποτίμησή τους μετά την αποτίμηση της αριστερής υποέκφρασης, αν από αυτή προκύπτει η τιμή της έκφρασης.
Για παράδειγμα, στη λογική έκφραση:

```
a || (x++ > y)
```

η υποέκφραση "x++ > y" δεν πρόκειται να αποτιμηθεί, αν η μεταβλητή a έχει λογική τιμή "Αληθής", οπότε η αριστερή υποέκφραση παρέχει την τιμή "Αληθής" για όλη την έκφραση. Ακόμα κι αν η δεξιά υποέκφραση παράγει κάποιο πλάγιο αποτέλεσμα, όπως συμβαίνει παραπάνω, αυτό δε θα αποθηκευτεί. Έτσι, αν στην παραπάνω έκφραση η μεταβλητή a έχει τιμή "Αληθής", η μεταβλητή x διατηρεί την προηγούμενη τιμή της. Αντίθετα, αν η μεταβλητή a έχει λογική τιμή "Ψευδής", η δεξιά υποέκφραση θα αποτιμηθεί, και η μεταβλητή x θα αυξηθεί κατά 1.
Παρόμοιο είναι το αποτέλεσμα της βραχυκύκλωσης στην εφαρμογή του λογικού τελεστή '&&' για την αντίστοιχη λογική τιμή "Ψευδής".

Στην αρχική μορφή του ενδιάμεσου κώδικα σε μορφή αφηρημένου συντακτικού δέντρου, και εφ' όσον οι εκφράσεις αναπαριστώνται με δέντρα εκφράσεων, δεν προβάλλεται η δυνατότητα διακοπής της αποτίμησης μιας λογικής έκφρασης. Η βραχυκύκλωση ενσωματώνεται στον κώδικα καθώς προχωράμε, με κατάλληλο σχήμα παραγωγής τελικού κώδικα, προς μια αναπαράσταση χαμηλότερου επιπέδου.
Σύμφωνα με τα παραπάνω, για την υλοποίηση της βραχυκύκλωσης για κάποια λογική πράξη, θα χρειαστεί να εισάγουμε στο αφηρημένο συντακτικό δέντρο έναν κόμβο ελέγχου ροής, ο οποίος θα εκτελεί άλμα προς κατάλληλο άλλο κόμβο του δέντρου, ανάλογα με την τιμή της αριστερής υποέκφρασης του κόμβου της λογικής πράξης.
Στο παράδειγμα που δώσαμε, αν η μεταβλητή a έχει λογική τιμή "Αληθής", ο κώδικάς μας πρέπει να εκτελεί άλμα στον κώδικα που ακολουθεί την αποτίμηση της λογικής έκφρασης, υπερπηδώντας τον κώδικα αποτίμησης της δεξιάς υποέκφρασης..
Ο κόμβος ελέγχου ροής όμως δεν αρκεί να εκτελεί άλμα υπό συνθήκη, αλλά πρέπει και να παράγει κάποια τιμή, αφού η αποτίμηση μιας έκφρασης πρέπει να ολοκληρώνεται μετά τη βραχυκύκλωση. Πιο συγκεκριμένα, το άλμα που εκτελείται σε μια βραχυκύκλωση πρέπει να μεταφέρει και την τιμή της αριστερής υποέκφρασης στον κόμβο προορισμού.
Έτσι, στο παραπάνω παράδειγμα, σε περίπτωση βραχυκύκλωσης η τιμή της μεταβλητής a πρέπει να αποτελέσει την τελική τιμή της λογικής έκφρασης. Αν μάλιστα αυτή είναι μέρος μεγαλύτερης έκφρασης, η τιμή αυτή θα χρησιμοποιηθεί για την αποτίμηση της υπόλοιπης έκφρασης.
Επομένως, ο κόμβος ελέγχου ροής που θα εισάγουμε πρέπει να λειτουργεί και ως τελεστής, ο οποίος εφαρμοζόμενος σε ένα τελούμενο-υποέκφραση του δέντρου παράγει ως αποτέλεσμα την τιμή αυτού, και ανάλογα με αυτή, εκτελεί άλμα σε κάποιον άλλο κόμβο του δέντρου.
Έχοντας εξασφαλίσει ότι οι νέοι κόμβοι παράγουν τιμές, μπορούμε να χρησιμοποιήσουμε στη συνέχεια τη γνωστή διαδικασία παραγωγής τελικού κώδικα από δέντρα εκφράσεων, όπου κάθε κόμβος αποτιμάται μετά την αποτίμηση των παιδιών του, και ο τελικός κώδικας που αντιστοιχεί στο δέντρο παράγεται με διαπέραση αυτού κατά βάθος, και από αριστερά προς τα δεξιά.

Το σχήμα παραγωγής τελικού κώδικα για λογικές εκφράσεις με βραχυκύκλωση εφαρμόζεται με δύο διαδοχικές διαπεράσεις του δέντρου έκφρασης, κατά βάθος και από αριστερά προς τα δεξιά.
Στην πρώτη διαπέραση γίνεται και η αρίθμηση των κόμβων του δέντρου με τη σειρά αποτίμησής τους. Ειδικότερα, για κάθε κόμβο λογικής πράξης:

(α) Επισκεπτόμαστε το αριστερό παιδί του κόμβου.
(β) Κατασκευάζουμε ένα νέο κόμβο τύπου άλματος με συνθήκη, στον οποίο συνδέουμε το παραπάνω παιδί. Το πεδίο προορισμού άλματος του κόμβου αυτού θα το συμπληρώσουμε στην επόμενη διαπέραση του δέντρου.
(γ) Αριθμούμε το νέο κόμβο.
(δ) Αντικαθιστούμε το αριστερό παιδί του αρχικού κόμβου με τον παραπάνω κόμβο, και συνεχίζουμε τη διαπέραση με επίσκεψη του δεξιού παιδιού του αρχικού κόμβου.

Με την αρίθμηση που κάναμε στην παραπάνω διαπέραση, το πεδίο προορισμού άλματος κάθε νέου κόμβου μπορεί να δείχνει στον κόμβο με αριθμό κατά 1 μεγαλύτερο από τον αριθμό κόμβου του πατέρα του, επειδή αυτός ο αριθμός αντιστοιχεί στον κόμβο που αποτιμάται αμέσως μετά τη λογική πράξη. Όμως, για να λάβουμε τιμή αποτελέσματος από τη λογική πράξη για κάθε περίπτωση τιμής της αριστερής υποέκφρασης, το πεδίο προορισμού θα πρέπει να δείχνει στον ίδιο τον πατέρα του κόμβου άλματος, ο οποίος σύμφωνα με τα παραπάνω είναι ο κόμβος της λογικής πράξης. Στην περίπτωση βραχυκύκλωσης, η τιμή της αριστερής υποέκφρασης είναι και το αποτέλεσμα της πράξης, ανεξάρτητα από την άλλη τιμή που συμμετέχει στην πράξη, ενώ στην περίπτωση πλήρους υπολογισμού, η τιμή της αριστερής υποέκφρασης είναι το ουδέτερο στοιχείο της πράξης, και το αποτέλεσμα καθορίζεται από την τιμή της δεξιάς υποέκφρασης.

Τη δεύτερη διαπέραση του δέντρου τη χρησιμοποιούμε, ώστε να καλύψουμε λογικές εκφράσεις με διαδοχικές εφαρμογές του ίδιου τελεστή, οπότε η βραχυκύκλωση επεκτείνεται σε πολλαπλούς κόμβους. Ειδικότερα, στη δεύτερη διαπέραση, μόλις βρούμε έναν κόμβο άλματος με συνθήκη, θα εφαρμόσουμε τον εξής αλγόριθμο:

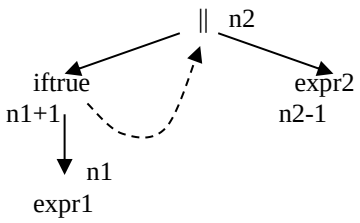
1. Θέσε σε μια μεταβλητή x τον αριθμό του πατέρα του κόμβου.
2. Όσο υπάρχει πατέρας του κόμβου με αριθμό x και είναι του ίδιου τύπου άλματος με συνθήκη, θέσε στη μεταβλητή x τον αριθμό του πατέρα του πατέρα του x.
3. Ο αριθμός που θα μπει στο πεδίο προορισμού άλματος του κόμβου θα είναι το x.

Η ορθότητα του παραπάνω αλγόριθμου στηρίζεται στην παρατήρηση ότι ο πατέρας του πατέρα του κόμβου είναι του ίδιου τύπου άλματος με συνθήκη, μόνο όταν έχουμε διαδοχική εφαρμογή του ίδιου λογικού τελεστή. Αυτό συμβαίνει επειδή (α) οι κόμβοι άλματος με συνθήκη εισάγονται μόνο για λογικούς τελεστές, και (β) οι δυαδικοί λογικοί τελεστές ‘||’ και ‘&&’ της C εξετάζουν διαφορετική συνθήκη στους κόμβους αυτούς.

Ας δούμε για παράδειγμα τη μορφή που λαμβάνει το δέντρο έκφρασης για την απλή έκφραση:

```
expr1 || expr2
```

μετά την εφαρμογή του παραπάνω σχήματος:



όπου σε κάθε κόμβο του δέντρου αναγράφεται και ο αριθμός του. Ο τελεστής άλματος με συνθήκη για τη λογική πράξη "Η" συμβολίζεται με το όνομα *iftrue* και ελέγχει το τελούμενό του για λογική τιμή "Αληθής". Αν αυτή είναι η τιμή της έκφρασης expr1, τότε η έκφραση expr2 δε θα αποτιμηθεί, και η λογική πράξη "Η" θα έχει επίσης αποτέλεσμα "Αληθής", διαφορετικά θα αποτιμηθεί η expr2 και η λογική πράξη "Η" θα έχει ως αποτέλεσμα την τιμή της expr2.

Η αρίθμηση των κόμβων του δέντρου έκφρασης, εκτός από τη σειρά αποτίμησης, μας δίνει και τους αριθμούς των προσωρινών μεταβλητών που χρησιμοποιούνται για την αποτίμηση της έκφρασης. Έτσι, ένας αλγόριθμος δέσμευσης καταχωρητών θα μας δώσει τους καταχωρητές της τελικής γλώσσας που θα αντιστοιχηθούν στις μεταβλητές αυτές. Μια απλή διαπέραση του τελικού αφηρημένου συντακτικού δέντρου θα μας δώσει και τον τελικό κώδικα, με μία ή δύο εντολές ανά κόμβο του δέντρου.
Ένας κώδικας MIPS που παράγεται για την προηγούμενη λογική έκφραση μπορεί να είναι ο ακόλουθος:

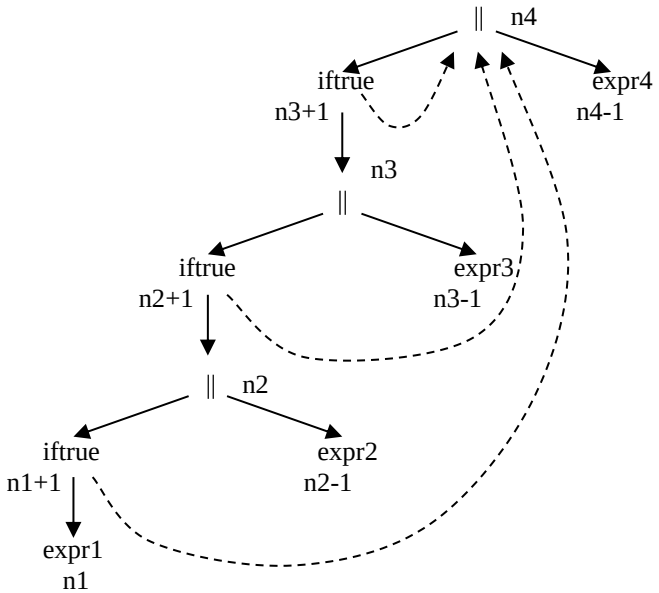
```
      <κώδικας αποτίμησης έκφρασης expr1, με αποτέλεσμα στον καταχωρητή $5>
      or $9,$5,$0
      bne $5,$0,L
      <κώδικας αποτίμησης έκφρασης expr2, με αποτέλεσμα στον καταχωρητή $8>
L:    or $6,$9,$8
```

όπου η πρώτη εντολή or προκύπτει από τον κόμβο άλματος, και αντιγράφει την τιμή του καταχωρητή $5 που αντιστοιχήθηκε στον κόμβο n1 στον καταχωρητή $9 που αντιστοιχήθηκε στον κόμβο άλματος, με την εντολή διακλάδωσης να ολοκληρώνει τον κώδικα του κόμβου άλματος. Παρατηρήστε ότι ο καταχωρητής $8 δε λαμβάνει τιμή αν η εντολή διακλάδωσης εκτελεί άλμα, αλλά αυτό συμβαίνει μόνο αν ο καταχωρητής $5 έχει μη μηδενική τιμή, οπότε η λογική πράξη θα έχει αποτέλεσμα "Αληθής", ανεξάρτητα από την τιμή του $8. Αν πάλι το άλμα δεν εκτελείται, αυτό θα συμβαίνει με μηδενική τιμή του $5, οπότε η λογική πράξη θα έχει ως αποτέλεσμα την τιμή του $8.

Ας δούμε τώρα τη μορφή του δέντρου για την πιο σύνθετη έκφραση:

```
expr1 || expr2 || expr3 || expr4
```

που λαμβάνουμε μετά την εφαρμογή του παραπάνω σχήματος:



Και στις τρεις περιπτώσεις εφαρμογής του τελεστή αυτού, το πεδίο προορισμού άλματος περιέχει δείκτη στον κόμβο με αριθμό n4.

Εδώ όμως δεν έχουμε ακόμα τελειώσει, γιατί ναι μεν εξασφαλίσαμε την αντιγραφή της τιμής μιας αριστερής υποέκφρασης στον κόμβο άλματος, αλλά τι γίνεται αν ο κόμβος προορισμού του άλματος δεν είναι ο ίδιος ο πατέρας του κόμβου άλματος; Μ' άλλα λόγια, με το παραπάνω σχήμα η εκτέλεση άλματος στην περίπτωση βραχυκύκλωσης αποτυγχάνει να μεταφέρει την τιμή της λογικής έκφρασης στον κόμβο λογικής πράξης που είναι ο προορισμός του άλματος. Ευτυχώς, το πρόβλημα αυτό μπορεί να αντιμετωπιστεί στη φάση δέσμευσης καταχωρητών.
Κατ' αρχήν παρατηρούμε ότι ο κόμβος άλματος με συνθήκη αντιγράφει την τιμή της αριστερής υποέκφρασης του αρχικού κόμβου λογικής πράξης στην προσωρινή μεταβλητή που του αντιστοιχείται. Επειδή δεν υπάρχει αλληλεπίδραση μεταξύ των δύο κόμβων, καθότι ο πρώτος είναι πατέρας του δεύτερου στο δέντρο έκφρασης, μπορούμε να υποχρεώσουμε τον αλγόριθμο δέσμευσης καταχωρητών να αποδώσει σε αυτούς τον ίδιο καταχωρητή – οπότε βέβαια η αντιγραφή δεν έχει πια νόημα. Παρόμοια, δεν υπάρχει αλληλεπίδραση μεταξύ του κόμβου άλματος με συνθήκη και του κόμβου αποτίμησης της λογικής έκφρασης, κι επομένως στους δύο αυτούς κόμβους μπορεί να αποδοθεί ο ίδιος καταχωρητής. Συνεχίζοντας, μπορούμε να αποδίδουμε τον ίδιο καταχωρητή, μέχρι να βρούμε τον κόμβο προορισμού του άλματος.
Η ορθότητα της αποτίμησης της λογικής έκφρασης στον τελικό κώδικα σε περίπτωση βραχυκύκλωσης εξασφαλίζεται με τον παραπάνω τρόπο, αφού έτσι ο καταχωρητής από τον οποίο αυτή θα ληφθεί θα έχει λάβει τιμή από τον κόμβο που αποτιμά την αντίστοιχη αριστερή υποέκφραση. Σε περίπτωση μη βραχυκύκλωσης από την άλλη μεριά, η τιμή του καταχωρητή συνεχίζει να είναι έγκυρη, ώστε να χρησιμοποιηθεί για την αποτίμηση της λογικής έκφρασης, επειδή ο καταχωρητής αυτός δε μπορεί να χρησιμοποιείται από άλλον κώδικα που μεσολαβεί, λόγω αλληλεπίδρασης με όλους τους κόμβους των αντίστοιχων δεξιών υποεκφράσεων.

Ένας κώδικας MIPS που παράγεται για την προηγούμενη λογική έκφραση, εφαρμόζοντας και τον παραπάνω περιορισμό στη δέσμευση καταχωρητών, μπορεί να είναι ο ακόλουθος:

```
      <κώδικας αποτίμησης έκφρασης expr1, με αποτέλεσμα στον καταχωρητή $5>
      bne $5,$0,L
      <κώδικας αποτίμησης έκφρασης expr2, με αποτέλεσμα στον καταχωρητή $8>
      or $5,$5,$8
      bne $5,$0,L
      <κώδικας αποτίμησης έκφρασης expr3, με αποτέλεσμα στον καταχωρητή $12>
      or $5,$5,$12
      bne $5,$0,L
      <κώδικας αποτίμησης έκφρασης expr4, με αποτέλεσμα στον καταχωρητή $6>
L:    or $5,$5,$6
```

Άσκηση 9-3:

Έστω η παρακάτω έκφραση της γλώσσας C:

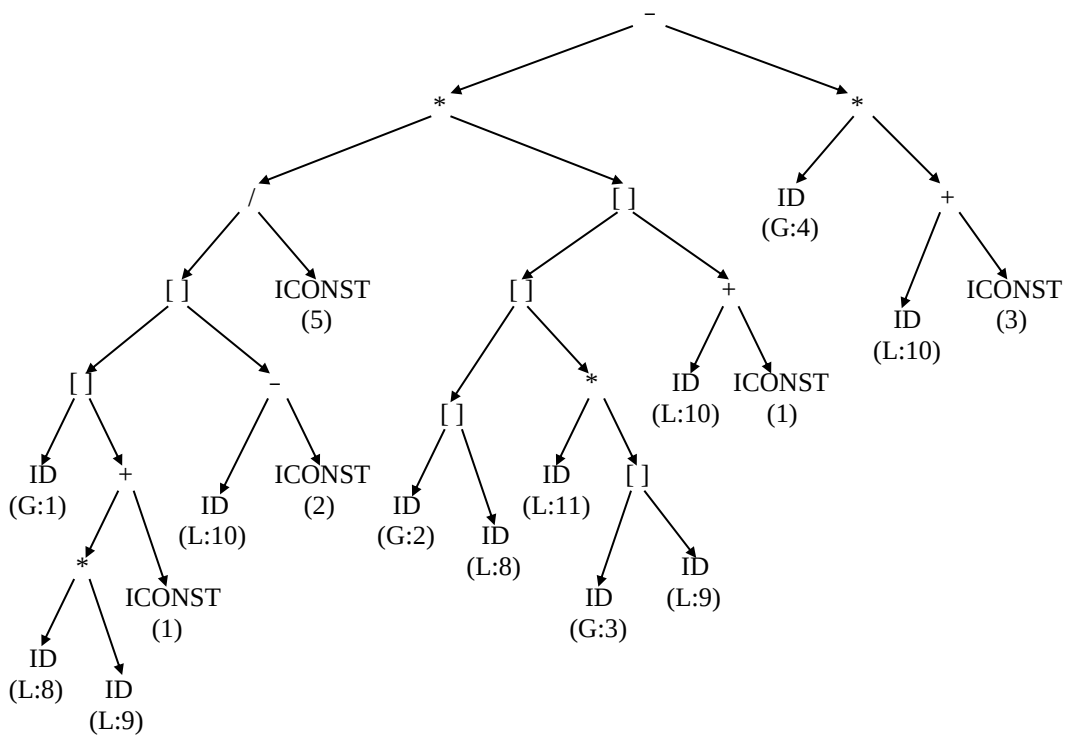
```
a[i*j+1][k-2]/5 * b[i][w*m[j]][k+1] - c * (k+3)
```

Λαμβάνοντας υπ' όψη ότι (α) οι πίνακες a και b είναι δηλωμένοι στην αρχή της εξωτερικής μονάδας ως int[1000][10] και int[100][1000][10], αντίστοιχα, (β) ο πίνακας m είναι δηλωμένος στην εξωτερική μονάδα, αμέσως μετά τους προηγούμενους, ως short[100], (γ) η καθολική μεταβλητή c είναι τύπου int και δηλωμένη αμέσως μετά τον πίνακα m, (δ) οι μεταβλητές στοίβας i, j και k είναι τύπου int, και έχουν δεσμεύσει διαδοχικές θέσεις στη στοίβα, ξεκινώντας από τη θέση 8, όταν οι προηγούμενες θέσεις έχουν αντιστοιχηθεί σε αντικείμενα μεγέθους ίσου με το μέγεθος του τύπου int, (ε) η μεταβλητή w είναι τύπου short και είναι τοποθετημένη στη στοίβα, αμέσως μετά τις προηγούμενες, (στ) η έκφραση είναι τύπου int, και (ζ) τα αντικείμενα των τύπων int και short έχουν μέγεθος 4 και 2 bytes, αντίστοιχα:
Α. Να δεσμεύσετε καταχωρητές για την έκφραση αυτή, σε ένα μεταγλωττιστή του οποίου ο ενδιάμεσος κώδικας έχει μορφή αφηρημένου συντακτικού δέντρου. Πριν τη δέσμευση καταχωρη-

τών, να φέρετε τον ενδιάμεσο κώδικα στην τελική του μορφή πριν την παραγωγή τελικού κώδικα MIPS.
Β. Πόσοι καταχωρητές απαιτούνται για την παραπάνω δέσμευση; Σκεφτείτε έναν τρόπο για να μειώσετε τον αριθμό καταχωρητών χωρίς διάχυση, εάν σε κάθε υποέκφραση η σειρά αποτίμησης των όρων δεν έχει σημασία. Επαναλάβετε τη δέσμευση με το νέο τρόπο.
Γ. Να δώσετε έναν τελικό κώδικα MIPS για την παραπάνω έκφραση, με βάση τη δέσμευση καταχωρητών που βρήκατε στο προηγούμενο ερώτημα.

Απάντηση

Α. Η αρχική μορφή του ενδιάμεσου κώδικα για την έκφρασή μας δίνεται παρακάτω:



Σύμφωνα με την εκφώνηση, η θέση της στοίβας, όπου τοποθετείται η μεταβλητή i, αντιστοιχεί στο δείκτη 8 του πίνακα δέσμευσης για το εγγράφημα δραστηριοποίησης της μονάδας, ενώ ακολουθούν οι μεταβλητές j, k και w. Άρα ο πίνακας δέσμευσης για τη μονάδα θα έχει τη μορφή:

```
{ ...; 8:int; 9:int; 10:int; 11:short; ... }
```

Στο χώρο στατικών δεδομένων, από την άλλη μεριά, τοποθετούνται πρώτες οι μεταβλητές a, b, m και c, με τη σειρά που αναγράφονται, κι επομένως στον πίνακα δέσμευσης της εξωτερικής μονάδας οι μεταβλητές αυτές αναφέρονται με τους δείκτες 1, 2, 3 και 4, αντίστοιχα:

```
{ 1:int[1000][10]; 2:int[100][1000][10]; 3:short[100]; 4:int; ...}
```

Για να προχωρήσουμε στην παραγωγή τελικού κώδικα, πρέπει να φέρουμε τον ενδιάμεσο κώδικα σε κατάλληλη μορφή για τη δέσμευση καταχωρητών. Η μορφή αυτή είναι η τελική μορφή ενδιάμεσου κώδικα, αφού δηλαδή (α) εφαρμόσουμε όποιες βελτιστοποιήσεις ενδιάμεσου κώδικα επιλέξουμε, (β) αναπτύξουμε όποιους κόμβους ενδιάμεσου κώδικα παράγουν πολλαπλές εντολές τελικού κώδικα, (γ) συμπτύξουμε όποια μέρη με πολλαπλές εντολές ενδιάμεσου κώδικα παράγουν απλές εντολές τελικού κώδικα, και (δ) μετατρέψουμε τους πίνακες δέσμευσης σε πίνακες διευθύνσεων στους χώρους δεδομένων. Δεν είναι απαραίτητο να κάνουμε τις μετατροπές αυτές με τη σειρά που δίνονται, ενώ οι (α), (β) και (γ) είναι πιθανό να γίνονται εναλλάξ.

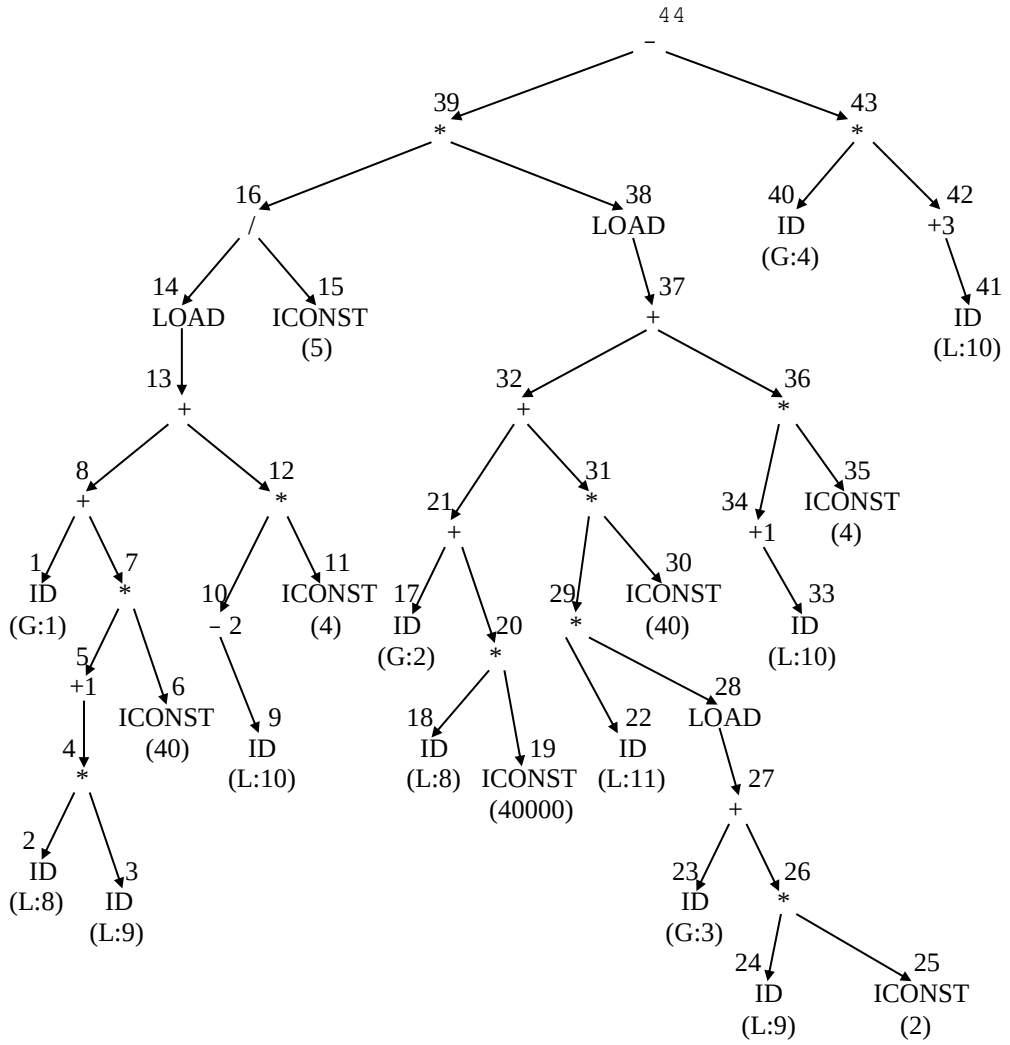
Στην προκειμένη περίπτωση δεν κάνουμε βελτιστοποιήσεις, κι έτσι θα προχωρήσουμε στις (β), (γ) και (δ) από τις παραπάνω μετατροπές. Εκτός από τη (δ), οι υπόλοιπες μετατροπές δεν είναι υποχρεωτικές για την παραγωγή τελικού κώδικα, είναι όμως επιθυμητές, γιατί οδηγούν σε πιο άμεση αντιστοίχηση κόμβων ενδιάμεσου κώδικα σε εντολές MIPS, κάτι που σημαίνει με μεγάλη πιθανότητα πιο αποδοτικό τελικό κώδικα.

Ανάμεσα στα μέρη του ενδιάμεσου κώδικα που δέχονται ανάπτυξη σε πολλαπλούς νέους κόμβους ενδιάμεσου κώδικα, εξέχουσα θέση κατέχουν οι επιλύσεις αναφορών σε στοιχεία πίνακα. Με τον όρο "επίλυση αναφοράς" ενός αντικειμένου του αρχικού προγράμματος εννοούμε τον υπολογισμό της διεύθυνσής του στο χώρο δεδομένων. Έτσι, στην ενδιάμεση γλώσσα της μορφής αφηρημένου συντακτικού δέντρου που χρησιμοποιούμε, οι επιλύσεις αναφορών σε στοιχεία πίνακα γίνονται με αντικατάσταση του κόμβου του τελεστή '[ ]' με όσους κόμβους απαιτούνται για την εύρεση της διεύθυνσης του στοιχείου που ζητείται.
Για παράδειγμα, η αναφορά σε ένα στοιχείο a[x][y] του πίνακα a του κώδικά μας, επιλύεται με αντικατάσταση του υποδέντρου του οποίου η ρίζα περιέχει τον δεξιότερο από τους διαδοχικούς τελεστές '[ ]', με ένα υποδέντρο που υπολογίζει την έκφραση "a + (x*10 + y)*4", όπου με a εννοούμε τη διεύθυνση βάσης του πίνακα, δηλαδή τη διεύθυνση του πρώτου στοιχείου του πίνακα, και 4 είναι το μέγεθος σε bytes του στοιχείου του a.
Αξίζει να σημειωθεί ότι αν η αρχική γλώσσα είχε ισχυρό σύστημα τύπων, τότε θα έπρεπε να προσθέσουμε και τον κώδικα που ελέγχει τις τιμές που έχουν οι δείκτες του πίνακα, για ανίχνευση πιθανού λάθους. Στη C που δεν έχει ισχυρό σύστημα τύπων δεν ελέγχονται οι τιμές των δεικτών, κι επομένως είναι δυνατό η τελική διεύθυνση που βρίσκει ο τελικός κώδικας να μην περιέχει στοιχείο του a.

Άλλα σημεία του αρχικού ενδιάμεσου κώδικα που αναπτύσσονται σε νέους κόμβους κώδικα και θα μας απασχολήσουν στην έκφραση που μας δόθηκε, είναι τα σημεία αποτίμησης δεξιάς προσπέλασης μεταβλητών. Αυτές γίνονται με ανάγνωση από τη μνήμη. Υποθέτοντας ότι τα φύλλα του συντακτικού δέντρου που αναφέρονται σε βαθμωτές μεταβλητές μπορούν να παραστήσουν τις δεξιές προσπελάσεις αυτών χωρίς κάποια μετατροπή, μας μένει να εισάγουμε κόμβους φόρτωσης για στοιχεία πίνακα.
Όσο αφορά σύμπτυξη κώδικα, χαρακτηριστική περίπτωση αποτελούν κόμβοι ενδιάμεσου κώδικα που αναπαριστούν σταθερές, όταν αυτές μπορούν να ενσωματωθούν σε κόμβους πράξεων. Για παράδειγμα, η αύξηση της τιμής μιας έκφρασης κατά μια σταθερά x μπορεί να γίνει με ένα μόνο κόμβο κώδικα, αντί για δύο.
Επειδή τόσο η ανάπτυξη όσο και η σύμπτυξη κώδικα στα σημεία που αναφέρθηκαν εξαρτώνται από την αρχιτεκτονική του τελικού κώδικα, για παράδειγμα από το μέγεθος σε bytes του βασικού τύπου ενός πίνακα, από τις δυνατότητες προσπέλασης μνήμης της αρχιτεκτονικής, ή από τις εντολές πράξεων της τελικής γλώσσας, δε μπορούν να γίνουν κατά την παραγωγή της αρχικής μορφής του ενδιάμεσου κώδικα, όταν θέλουμε ο ενδιάμεσος κώδικας να είναι ανεξάρτητος από την αρχιτεκτονική της τελικής γλώσσας μετάφρασης.

Πριν προχωρήσουμε στην τελική μορφή ενδιάμεσου κώδικα για την έκφρασή μας, θα μετατρέψουμε τους πίνακες δέσμευσης για τις θέσεις που μας απασχολούν σε πίνακες διευθύνσεων. Έτσι, στο χώρο στατικών δεδομένων, οι διευθύνσεις – σχετικά με την αρχή του – θα δίνονται από τον πίνακα {0, 40000, 4040000, 4040200, …}, που αντικαθιστά τον αρχικό πίνακα δέσμευσης. Στο εγγράφημα δραστηριοποίησης της μονάδας που περιέχει την έκφραση, ο αντίστοιχος πίνακας θα είναι {…, 28, 32, 36, 40, …}, όπου η διεύθυνση 28 σχετικά με την κορυφή της στοίβας περιέχει τη μεταβλητή i.

Εφ' όσον η τελική αρχιτεκτονική είναι αρχιτεκτονική MIPS, οι αναπτύξεις και συμπτύξεις κώδικα που μπορεί αυτή να επιτρέψει για την έκφρασή μας, δίνει την τελική μορφή ενδιάμεσου κώδικα που ακολουθεί.



Μπορούμε στον παραπάνω κώδικα να δούμε (α) τις πράξεις που γίνονται για την επίλυση αναφορών σε στοιχεία πίνακα, (β) τις προσθήκες κόμβων με τον ψευδοτελεστή LOAD για φόρτωση της τιμής των στοιχείων πίνακα, όταν αυτά είναι τιμές δεξιάς προσπέλασης, και (γ) τις συμπτύξεις κόμβων σταθερών με κόμβους πρόσθεσης ή αφαίρεσης, αφού η αρχιτεκτονική MIPS επιτρέπει την κωδικοποίηση των σταθερών μαζί με τις εντολές των αντίστοιχων πράξεων.

Με την παραγωγή της τελικής μορφής του ενδιάμεσου κώδικα, κάθε κόμβος αυτού αντιστοιχεί σε μια εντολή της τελικής γλώσσας της μετάφρασης.[4] Έτσι, στον ενδιάμεσο κώδικα που αποτιμά μια έκφραση, όπως αυτή που μας δίνεται, κάθε κόμβος παράγει μια τιμή που θεωρούμε ότι αποθηκεύεται σε μια προσωρινή μεταβλητή. Σ' αυτή την περίπτωση, μια απλή μέθοδος δέσμευσης καταχωρητών είναι με δέσμευση ενός νέου καταχωρητή για κάθε προσωρινή μεταβλητή. Προφανώς αυτή η μέθοδος μπορεί να εφαρμοστεί, μόνο αν οι διαθέσιμοι καταχωρητές είναι τουλάχιστον τόσοι, όσες και οι προσωρινές μεταβλητές, δηλαδή όσοι και οι κόμβοι του αφηρημένου συντακτικού δέντρου.
Επειδή όμως δεν υπάρχει αρχιτεκτονική MIPS που να διαθέτει 44 καταχωρητές, όσοι είναι οι κόμβοι του συγκεκριμένου κώδικα, θα προχωρήσουμε σε δέσμευση καταχωρητών για τον κώδικα αυτόν με τη μέθοδο του χρωματισμού του γράφου αλληλεπιδράσεων. Υποθέτοντας ότι οι εντολές του τελικού κώδικα παράγονται με διαπέραση του αφηρημένου συντακτικού δέντρου κατά βάθος, και από αριστερά προς τα δεξιά, η σειρά επίσκεψης των κόμβων του

[4] Παρ' όλη την προσπάθεια που καταβάλλεται γι' αυτό, κάποιες φορές ο μεταγλωττιστής αναγκάζεται να χρησιμοποιήσει περισσότερες από μία εντολές τελικού κώδικα για έναν κόμβο ενδιάμεσου κώδικα.

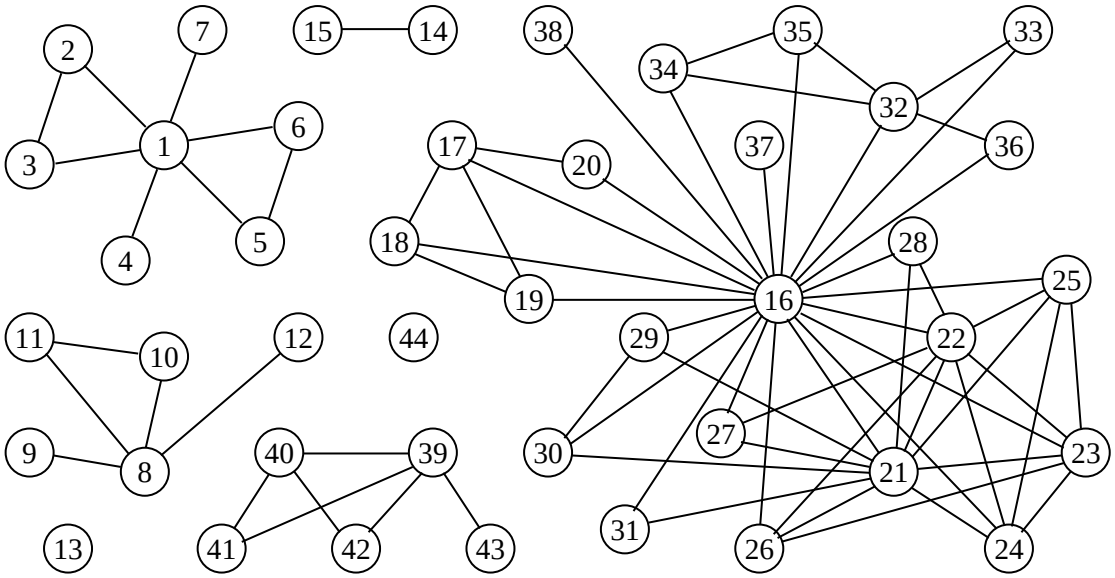
παραπάνω ενδιάμεσου κώδικα – και επομένως η σειρά των αντίστοιχων εντολών του τελικού κώδικα – φαίνεται με την αρίθμηση που κάναμε στους κόμβους του.
Η αρίθμηση αυτή δίνει έμμεσα τη διάρκεια ζωής της τιμής του κάθε κόμβου, επειδή αν δεν απαλείψουμε κοινές υποεκφράσεις, η τιμή ενός κόμβου χρησιμοποιείται ακριβώς μια φορά μετά τον υπολογισμό της. Η διάρκεια ζωής της τιμής ενός κόμβου θα ισούται με τη διαφορά των αριθμών των ετικετών αυτού από του πατέρα του, αυξημένη κατά 1, ξεκινώντας από τη στιγμή παραγωγής της τιμής του, αφού ο πατέρας του είναι και ο μοναδικός κόμβος που τη χρησιμοποιεί. Για παράδειγμα, η διάρκεια ζωής της τιμής της μεταβλητής w που διαβάζεται στον κόμβο 22 θα είναι 29-22+1=8 εντολές, ξεκινώντας από την εντολή που αντιστοιχεί στον κόμβο με αριθμό 22 και τελειώνοντας στην εντολή που αντιστοιχεί στον κόμβο με αριθμό 29.

Έχοντας λοιπόν υπολογίσει τις διάρκειες ζωής των τιμών όλων των κόμβων του κώδικα, μπορούμε να βρούμε τις αλληλεπιδράσεις μεταξύ των κόμβων. Ένας κόμβος αλληλεπιδρά με όλους τους κόμβους με τιμές που έχουν κάποια κοινή διάρκεια ζωής με τη διάρκεια ζωής της τιμής του.
Για να κατασκευάσουμε το γράφο αλληλεπιδράσεων, μπορούμε να αποδείξουμε ότι με βάση τα παραπάνω, ένας κόμβος θα αλληλεπιδρά με εκείνους τους κόμβους που έχουν αριθμό ετικέτας μεταξύ του αριθμού ετικέτας του κόμβου και του αριθμού ετικέτας του πατέρα του, και ότι αυτό αρκεί για την κατασκευή του ζητούμενου γράφου, με διαδοχική εφαρμογή του σε όλους τους κόμβους του κώδικα:

Κατ' αρχήν, είναι φανερό ότι ένας κόμβος αλληλεπιδρά με τους παραπάνω, αφού από την αρίθμηση των κόμβων προκύπτει ότι οι τιμές τους έχουν διάρκειες ζωής που περιέχονται στη διάρκεια ζωής της τιμής του.
Για να δούμε ότι αυτό αρκεί για την κατασκευή του γράφου αλληλεπιδράσεων, πρέπει να δείξουμε ότι η διαδοχική εφαρμογή του σε όλους τους κόμβους του ενδιάμεσου κώδικα βρίσκει όλες τις αλληλεπιδράσεις μεταξύ των κόμβων.
Έστω λοιπόν ότι εξετάζουμε τον κόμβο με αριθμό ετικέτας n. Οι κόμβοι με αριθμό ετικέτας μεγαλύτερο από τον αριθμό ετικέτας του πατέρα του κόμβου δε μπορούν να αλληλεπιδρούν με αυτόν, αφού η διάρκεια ζωής της τιμής του κόμβου τελειώνει στον κόμβο πατέρα του, ενώ η διάρκεια ζωής των τιμών αυτών ξεκινάει από αυτούς. Ακόμα, σύμφωνα με τον κανόνα που εφαρμόζουμε, ο κόμβος με ετικέτα n αλληλεπιδρά με όλους τους κόμβους με αριθμό ετικέτας μεταξύ του n και του αριθμού ετικέτας του πατέρα του.
Αρκεί να δείξουμε ότι όλες οι αλληλεπιδράσεις του κόμβου με ετικέτα n με κόμβους που έχουν αριθμό ετικέτας μικρότερο από n βρίσκονται κατά την εξέταση των κόμβων αυτών, και δε μας διαφεύγουν. Έστω m ο αριθμός ετικέτας ενός τέτοιου κόμβου. Ο πατέρας του κόμβου αυτού δε μπορεί να έχει ετικέτα με αριθμό μικρότερο από n, διαφορετικά η διάρκεια ζωής της τιμής του θα τελείωνε πριν από τον κόμβο με ετικέτα n, και δε θα υπήρχε αλληλεπίδραση. Αλλά αν ο πατέρας του κόμβου με ετικέτα m έχει ετικέτα με αριθμό μεγαλύτερο ή ίσο με n, η αλληλεπίδραση σημειώνεται κατά την εξέταση του κόμβου με αριθμό ετικέτας m.□

Πριν προχωρήσουμε στην κατασκευή του γράφου αλληλεπιδράσεων, παρατηρούμε ότι η αρχιτεκτονική MIPS εξαλείφει αλληλεπιδράσεις μεταξύ ενός κόμβου και του πατέρα του, επειδή η χρήση της τιμής του κόμβου προηγείται της αποθήκευσης της τιμής του πατέρα του. Έτσι, η διάρκεια ζωής της τιμής του κόμβου τελειώνει πριν αρχίσει η διάρκεια ζωής της τιμής του πατέρα του, γεγονός που επιτρέπει τη δέσμευση του ίδιου καταχωρητή για τους δύο κόμβους.

Εξετάζοντας διαδοχικά τους κόμβους του ενδιάμεσου κώδικα που έχουμε, και λαμβάνοντας υπ' όψη την παραπάνω παρατήρηση, κατασκευάζουμε το γράφο αλληλεπιδράσεων της επόμενης σελίδας.
Παρατηρούμε ότι ο γράφος αυτός αποτελείται από περισσότερα του ενός συνεκτικά τμήματα, ενώ κάποιοι κόμβοι δε συνδέονται με άλλους. Τέτοιος κόμβος είναι τουλάχιστον ο κόμβος ρίζα του δέντρου του ενδιάμεσου κώδικα – εδώ ο κόμβος με ετικέτα 44, που προφανώς δεν αλληλεπιδρά με κανένα άλλο κόμβο του δέντρου.



Πρέπει να σημειωθεί ότι ο παραπάνω γράφος αρκεί για τη δέσμευση καταχωρητών, όταν όλοι οι διαθέσιμοι καταχωρητές είναι ισοδύναμοι, και μπορούν να αποδοθούν σε οποιονδήποτε κόμβο. Αν υπάρχουν περιορισμοί στη χρήση των καταχωρητών, τότε επεκτείνουμε τον πιο πάνω γράφο με κόμβους καταχωρητών, σημειώνοντας τους όποιους περιορισμούς σαν αλληλεπιδράσεις μεταξύ αυτών και των υπόλοιπων κόμβων. Οι κόμβοι καταχωρητών αποτελούν έναν πλήρη υπογράφο (κλίκα) στο γράφο αλληλεπιδράσεων, αφού προφανώς δύο τέτοιοι κόμβοι δε μπορούν να έχουν ίδιο χρώμα.
Εναλλακτικά, αν οι καταχωρητές της τελικής αρχιτεκτονικής χωρίζονται σε περισσότερες από μία κατηγορίες ισοδύναμων καταχωρητών, έτσι ώστε ένας κόμβος ενδιάμεσου κώδικα να μπορεί να αποδοθεί σε μία και μόνο κατηγορία, αντί για προσθήκη κόμβων καταχωρητών, είναι πιο εύκολο να απαλειφθούν από το γράφο αλληλεπιδράσεων όλες οι αλληλεπιδράσεις μεταξύ κόμβων διαφορετικής κατηγορίας. Για παράδειγμα, κάτι τέτοιο μπορεί να γίνει στην αρχιτεκτονική MIPS, ώστε να διαχωριστεί η δέσμευση καταχωρητών σταθερής από τη δέσμευση καταχωρητών κινητής υποδιαστολής.

Η δέσμευση καταχωρητών για τον κώδικά μας μπορεί τώρα να ολοκληρωθεί με χρωματισμό του γράφου αλληλεπιδράσεων. Χρωματισμός ενός γράφου είναι η απόδοση χρωμάτων στους κόμβους του, έτσι ώστε να μην υπάρχουν δύο γειτονικοί κόμβοι με το ίδιο χρώμα. Σα χρώμα κόμβου εννοούμε μια τιμή από ένα σύνολο τιμών που καθορίζουν κάποιο χαρακτηριστικό του κόμβου. Στην περίπτωσή μας, η τιμή που αποδίδουμε είναι ο αριθμός καταχωρητή.
Το πρόβλημα του χρωματισμού ενός γράφου είναι η εύρεση ενός χρωματισμού με τον ελάχιστο αριθμό χρωμάτων, που λέγεται χρωματικός αριθμός του γράφου. Μια απλή ευριστική τεχνική χρωματισμού γράφου που βρίσκει ένα χρωματισμό με μικρό αριθμό χρωμάτων είναι με διαδοχική επίσκεψη των κόμβων του σε φθίνουσα σειρά βαθμού[5] τους, και με απόδοση σε κάθε κόμβο ενός χρώματος που δεν έχει δοθεί σε κανέναν από τους γειτονικούς του κόμβους.
Η τεχνική αυτή στηρίζεται στο γεγονός ότι η καθυστερημένη επίσκεψη ενός κόμβου είναι πιο πιθανό να οδηγήσει σε απόδοση νέου χρώματος στον κόμβο, όταν αυτός έχει μεγάλο βαθμό – άρα πολλούς γειτονικούς κόμβους, παρά όταν έχει μικρό βαθμό. Άρα, αν επισκεφτούμε πρώτα τους κόμβους μεγάλου βαθμού, μειώνουμε την πιθανότητα να λάβουμε ένα χρωματισμό με μεγάλο αριθμό χρωμάτων.

Η πιο πάνω τεχνική περιγράφεται αλγοριθμικά ως εξής:

1. Ταξινόμησε τους κόμβους σε φθίνουσα σειρά βαθμού.
2. Για κάθε κόμβο από την ταξινομημένη λίστα, έστω n:
2.1. Βρες κάποιο χρώμα που να μην έχει αποδοθεί σε κάποιο κόμβο γειτονικό του n.

[5] Βαθμός ενός κόμβου είναι ο αριθμός των κόμβων με τους οποίους συνδέεται.

2.2. Απόδωσε το χρώμα αυτό στον κόμβο n.

Εφαρμόζοντας τον πιο πάνω αλγόριθμο, ταξινομούμε τους κόμβους του γράφου αλληλεπιδράσεων που κατασκευάσαμε, ως εξής:

16, 21, 22, 1, 23, 24, 25, 32, 8, 17, 26, 39, 18, 19, 27, 28, 29, 30, 34, 35, 40, 2, 3, 5, 6, 10, 11, 20, 31, 33, 36, 41, 42, 4, 7, 9, 12, 14, 15, 37, 38, 43, 13, 44.

Στη συνέχεια ακολουθούμε το βήμα 2 του αλγόριθμου για καθένα από τους 44 κόμβους:

1. Αποδίδουμε αυθαίρετα στον κόμβο με ετικέτα 16 ένα χρώμα, πχ $10.
2. Αποδίδουμε στον κόμβο 21 ένα δεύτερο χρώμα, πχ $11.
3. Αποδίδουμε στον κόμβο 22 ένα τρίτο χρώμα, πχ $12, διότι αλληλεπιδρά και με τους δύο προηγούμενους κόμβους.
4. Αποδίδουμε στον κόμβο 1 ένα οποιοδήποτε από τα προηγούμενα χρώματα, έστω το $10.
5. Αποδίδουμε στον κόμβο 23 ένα τέταρτο χρώμα, πχ $13, διότι αλληλεπιδρά με τους κόμβους 16, 21 και 22, στους οποίους έχουν αποδοθεί τα τρία πρώτα χρώματα.
6. Αποδίδουμε στον κόμβο 24 ένα πέμπτο χρώμα, πχ $14, επειδή αλληλεπιδρά με τους κόμβους 16, 21, 22 και 23, στους οποίους έχουν αποδοθεί τα τέσσερα πρώτα χρώματα.
7. Αποδίδουμε στον κόμβο 25 ένα έκτο χρώμα, πχ $15, επειδή αλληλεπιδρά με τους κόμβους 16, 21, 22, 23 και 24, στους οποίους έχουν αποδοθεί τα πέντε πρώτα χρώματα.
8. Αποδίδουμε στον κόμβο 32 ένα χρώμα διαφορετικό από το χρώμα του κόμβου 16, με τον οποίο αλληλεπιδρά, πχ το $11.
9. Αποδίδουμε στον κόμβο 8 οποιοδήποτε χρώμα, πχ το $10.
10. Αποδίδουμε στον κόμβο 17 το χρώμα $11, με το ίδιο σκεπτικό, όπως για τον κόμβο 32.
11. Αποδίδουμε στον κόμβο 26 ένα χρώμα διαφορετικό από τα $10, $11, $12 και $13, έστω το $14.
12. Αποδίδουμε στον κόμβο 39 οποιοδήποτε χρώμα, έστω το $10.
13. Αποδίδουμε στον κόμβο 18 το $12, διότι αλληλεπιδρά με τους κόμβους 16 και 17.
14. Αποδίδουμε στον κόμβο 19 το $13, επειδή αλληλεπιδρά με τους κόμβους 16, 17 και 18.
15. Αποδίδουμε στον κόμβο 27 επίσης το χρώμα $13, διότι αλληλεπιδρά με τους κόμβους 16, 21 και 22.
16. Αποδίδουμε στον κόμβο 28 το $13, για τον ίδιο λόγο.
17. Αποδίδουμε στον κόμβο 29 το $12, διότι αλληλεπιδρά με τους κόμβους 16 και 21.
18. Αποδίδουμε στον κόμβο 30 το $13, διότι αλληλεπιδρά με τους κόμβους 16, 21 και 29.
19. Αποδίδουμε στον κόμβο 34 ένα χρώμα διαφορετικό από τα $10 και $11, έστω το $12.
20. Αποδίδουμε στον κόμβο 35 ένα χρώμα διαφορετικό από τα $10, $11 και $12, πχ το $13.
21. Αποδίδουμε στον κόμβο 40 το χρώμα $11, διότι αλληλεπιδρά με τον κόμβο 39.
22. Αποδίδουμε στον κόμβο 2 το $11, επειδή αλληλεπιδρά με τον κόμβο 1.
23. Αποδίδουμε στον κόμβο 3 το $12, επειδή αλληλεπιδρά με τους κόμβους 1 και 2.
24. Αποδίδουμε στον κόμβο 5 το $11.
25. Αποδίδουμε στον κόμβο 6 το $12.
26. Αποδίδουμε στον κόμβο 10 το $11.
27. Αποδίδουμε στον κόμβο 11 το $12.
28. Αποδίδουμε στον κόμβο 20 το $12, διότι αλληλεπιδρά με τους κόμβους 16 και 17.
29. Αποδίδουμε στον κόμβο 31 το $12, διότι αλληλεπιδρά με τους κόμβους 16 και 21.
30. Αποδίδουμε στον κόμβο 33 το $12, επειδή αλληλεπιδρά με τους κόμβους 16 και 32.
31. Αποδίδουμε στον κόμβο 36 το $12 για τον ίδιο λόγο.
32. Αποδίδουμε στον κόμβο 41 το $12.
33. Αποδίδουμε στον κόμβο 42 το $12.
34. Αποδίδουμε στον κόμβο 4 το χρώμα $11.
35. Αποδίδουμε στον κόμβο 7 το $11.
36. Αποδίδουμε στον κόμβο 9 το $11.
37. Αποδίδουμε στον κόμβο 12 το $11.
38. Αποδίδουμε στον κόμβο 14 το χρώμα $10.
39. Αποδίδουμε στον κόμβο 15 το $11.
40. Αποδίδουμε στον κόμβο 37 το $11.

41. Αποδίδουμε στον κόμβο 38 το $11.
42. Αποδίδουμε στον κόμβο 43 το $11.
43. Αποδίδουμε στον κόμβο 13 το $10.
44. Αποδίδουμε στον κόμβο 44 το $10.

Στην πιο πάνω εφαρμογή του αλγόριθμου χρωματισμού του γράφου αλληλεπιδράσεων, κάθε φορά που είχαμε επιλογή χρώματος, αποδώσαμε χρώματα με αύξουσα σειρά από το $10. Κάτι τέτοιο δεν είναι αναγκαίο, αλλά διευκολύνει την εφαρμογή.

Η τελική αρχιτεκτονική είναι πιθανό να έχει λιγότερους καταχωρητές από όσους μας δίνει ο χρωματισμός, είτε επειδή αυτός δεν είναι επιτυχημένος, είτε επειδή ο χρωματικός αριθμός του γράφου αλληλεπιδράσεων είναι μεγαλύτερος από τον αριθμό καταχωρητών αυτής.
Τότε, επιλέγουμε την προσωρινή μεταβλητή με τη μεγαλύτερη διάρκεια ζωής και τη *διαχέουμε* στη μνήμη, δηλαδή της αποδίδουμε κάποια μεταβλητή στοίβας, σε μια θέση που ακολουθεί τις θέσεις που έχουν δεσμευτεί μέχρι τώρα. Αυτό ουσιαστικά ισοδυναμεί με διάσπαση της έκφρασης που μεταφράζουμε σε δύο εκφράσεις, από τις οποίες η πρώτη υπολογίζει την τιμή της προσωρινής μεταβλητής και την αποθηκεύει στη στοίβα, ενώ η δεύτερη χρησιμοποιεί την τιμή αυτή, διαβάζοντάς την από τη στοίβα.
Μετά από αυτή τη μετατροπή απαλείφουμε από το γράφο αλληλεπιδράσεων όσες αλληλεπιδράσεις σχετίζονταν με τον κόμβο της μεταβλητής αυτής, και επαναλαμβάνουμε το χρωματισμό. Ας σημειωθεί ότι μετά από μια τέτοια απαλοιφή, δεν είναι δυνατό να υπάρχει συνεκτικό τμήμα στο γράφο που να περιέχει κόμβους και από τις δύο εκφράσεις, γεγονός που είναι αναμενόμενο, εφ' όσον το δέντρο ενδιάμεσου κώδικα έχει διασπαστεί σε δύο ασύνδετα δέντρα.
Η παραπάνω διαδικασία επαναλαμβάνεται όσες φορές χρειάζεται, ώσπου να βρούμε κάποιο χρωματισμό με τόσους καταχωρητές, όσους διαθέτει η αρχιτεκτονική.

Ας σημειωθεί ότι αν έχουμε κάνει εύρεση κοινών υποεκφράσεων πριν τη δέσμευση καταχωρητών, οπότε ο ενδιάμεσος κώδικας πάνω στον οποίο δουλεύουμε έχει μορφή που πιθανά δεν είναι δεντρική, πολλά από όσα αναφέραμε μέχρι τώρα δε θα ισχύουν επακριβώς, αλλά θα πρέπει να προσαρμοστούν στη μορφή αυτή. Έτσι, οι διάρκειες ζωής των προσωρινών μεταβλητών θα είναι πιθανά μεγαλύτερες, ώστε να λαμβάνουν υπ' όψη όλους τους πατέρες του κάθε κόμβου, με αποτέλεσμα ο γράφος αλληλεπιδράσεων να είναι πιθανά πιο πολύπλοκος. Ο αλγόριθμος της δέσμευσης καταχωρητών με χρωματισμό του γράφου αλληλεπιδράσεων θα είναι ο ίδιος, αλλά σε περίπτωση που απαιτηθεί διάχυση, η επιλογή της προσωρινής μεταβλητής για διάχυση θα λαμβάνει υπ' όψη εκτός από τη διάρκεια ζωής, και τον αριθμό των πατέρων του αντίστοιχου κόμβου, ώστε να αποφευχθεί μεγάλος αριθμός προσπελάσεων μνήμης, ενώ η διάχυση δε θα οδηγεί απαραίτητα σε διάσπαση του αρχικού κώδικα σε δύο ασύνδετα μέρη.

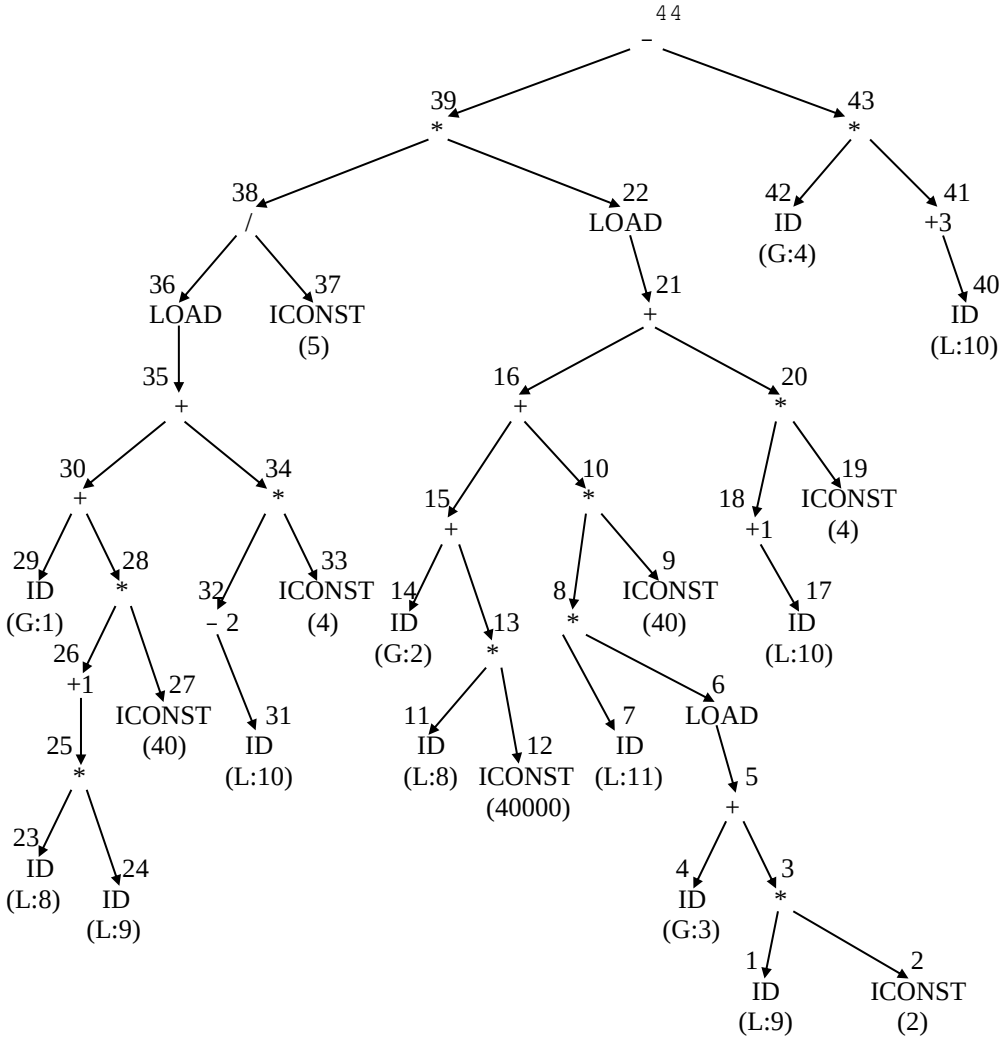
Β. Ο πιο πάνω χρωματισμός του γράφου αλληλεπιδράσεων μας έδωσε 6 χρώματα, δηλαδή 6 καταχωρητές για τον τελικό κώδικα που αποτιμά την έκφραση που μας δόθηκε, όπως δείχνει και ο πιο κάτω πίνακας:

Καταχωρητής	Κόμβοι
$10	1, 8, 13, 14, 16, 39, 44
$11	2, 4, 5, 7, 9, 10, 12, 15, 17, 21, 32, 37, 38, 40, 43
$12	3, 6, 11, 18, 20, 22, 29, 31, 33, 34, 36, 41, 42
$13	19, 23, 27, 28, 30, 35
$14	24, 26
$15	25

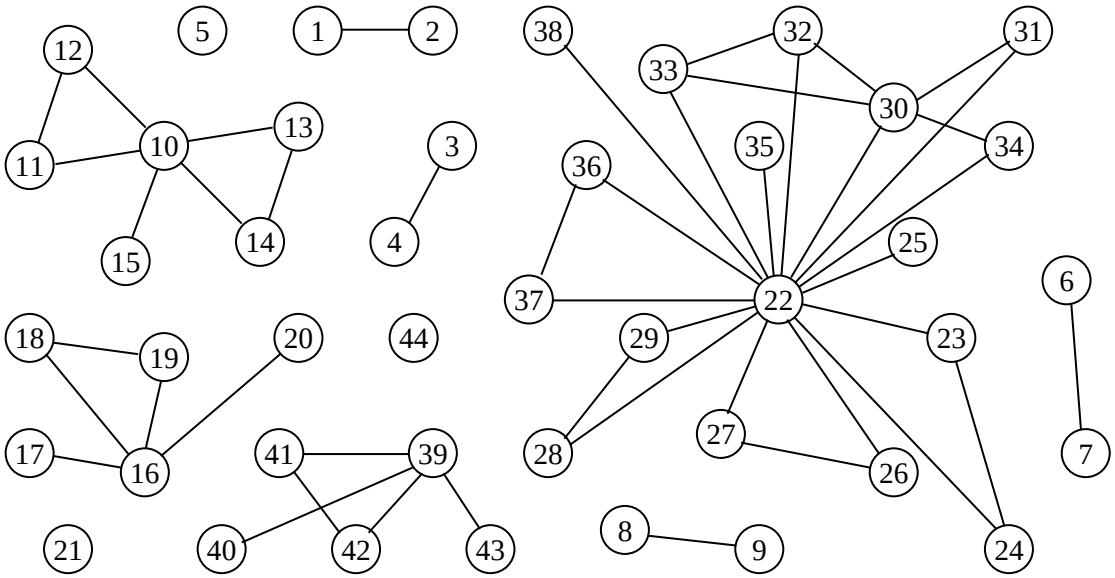
Για να μπορέσουμε να βρούμε κάποια δέσμευση καταχωρητών με λιγότερους από 6 καταχωρητές, πρέπει να ανακατασκευάσουμε το γράφο αλληλεπιδράσεων, ώστε να χρωματίζεται με

λιγότερα από 6 χρώματα. Κάτι τέτοιο μπορεί φυσικά να γίνει με τη μέθοδο της διάχυσης. Εδώ θα εξετάσουμε αν μπορεί να γίνει χωρίς διάχυση, με την προϋπόθεση ότι η σειρά αποτίμησης των όρων κάθε υποέκφρασης δεν έχει σημασία.
Αν λοιπόν η σειρά αποτίμησης δεν έχει σημασία, μπορούμε να στηριχθούμε στην πιο κάτω παρατήρηση, για να κατασκευάσουμε το νέο γράφο αλληλεπιδράσεων: Όποιο παιδί ενός κόμβου αποτιμηθεί πρώτο, η διάρκεια ζωής της τιμής του θα καλύπτει όλο το διάστημα αποτίμησης του άλλου παιδιού του κόμβου. Αν λοιπόν επιλέγουμε να αποτιμάμε πρώτα το παιδί με το μεγαλύτερο διάστημα αποτίμησης, η διάρκεια ζωής της τιμής του είναι μικρότερη από αυτή που θα είχε η τιμή του άλλου, αν το αποτιμούσαμε πρώτο.
Έστω για παράδειγμα το υποδέντρο του ενδιάμεσου κώδικα με ρίζα τον κόμβο με ετικέτα 8. Αν ο κόμβος 1, που είναι το αριστερό παιδί του 8, αποτιμηθεί πρώτος, η τιμή του έχει διάρκεια ζωής 8 εντολές, όσες χρειάζεται ο κόμβος 7 για την αποτίμησή του, συν την εντολή που αντιστοιχεί στην αποτίμηση του ίδιου του κόμβου 8. Αν ο κόμβος 7, που είναι το δεξί παιδί του 8, αποτιμηθεί πρώτος, η τιμή του έχει διάρκεια ζωής 2 μόλις εντολές, την εντολή που αντιστοιχεί στον κόμβο 1 και την εντολή που αντιστοιχεί στον κόμβο 8.
Στην πράξη, το διάστημα αποτίμησης ενός υποδέντρου του ενδιάμεσου κώδικα θα είναι ίσο με τον αριθμό των κόμβων του. Έτσι, μπορούμε να καθορίσουμε τη σειρά αποτίμησης των κόμβων του ενδιάμεσου κώδικα, μετρώντας τον αριθμό απογόνων κάθε κόμβου, και στη συνέχεια αριθμώντας τους κόμβους, με επιλογή σαν πρώτου μεταξύ δύο παιδιών ενός κόμβου, εκείνου που έχει το μεγαλύτερο αριθμό απογόνων.

Αριθμούμε λοιπόν ξανά τους κόμβους του ενδιάμεσου κώδικα, τον οποίο επαναλαμβάνουμε στη συνέχεια με τη νέα αρίθμηση:



Για να δούμε αν η νέα αρίθμηση οδηγεί σε δέσμευση καταχωρητών με λιγότερους καταχωρητές, επαναλαμβάνουμε τη διαδικασία που ακολουθήσαμε πιο πάνω. Ο γράφος αλληλεπιδράσεων βρίσκεται όπως προηγουμένως, μια που ο τρόπος κατασκευής του δεν επηρεάζεται από τη σειρά αρίθμησης, όσο αυτή γίνεται από κάτω προς τα επάνω. Έτσι, καταλήγουμε στον παρακάτω γράφο:



Η δέσμευση καταχωρητών που παίρνουμε με το χρωματισμό του νέου γράφου αλληλεπιδράσεων δίνεται στον πίνακα που ακολουθεί, και βλέπουμε ότι απαιτεί 4 καταχωρητές, κατά 2 λιγότερους από προηγουμένως.

Καταχωρητής	Κόμβοι
$10	1, 3, 5, 6, 8, 10, 16, 21, 22, 39, 44
$11	2, 4, 7, 9, 11, 13, 15, 17, 18, 20, 23, 25, 26, 28, 30, 35, 36, 38, 40, 41, 43
$12	12, 14, 19, 24, 27, 29, 31, 32, 34, 37, 42
$13	33

Γ. Όπως είπαμε νωρίτερα, η παραγωγή τελικού κώδικα γίνεται με τη σειρά αρίθμησης των κόμβων του τελικού αφηρημένου συντακτικού δέντρου.

Για κάθε εσωτερικό κόμβο του δέντρου, η εντολή τελικού κώδικα που παράγεται, χρησιμοποιεί σαν τελούμενα εισόδου τους καταχωρητές όπου βρίσκονται οι τιμές των παιδιών του, και σαν τελούμενο εξόδου τον καταχωρητή που δεσμεύτηκε γι' αυτόν. Ειδικά για τα τελούμενα εισόδου, αυτά αναγράφονται με τη σειρά που βρίσκονται στο δέντρο οι αντίστοιχοι κόμβοι, δηλαδή πρώτα ο καταχωρητής του αριστερού και μετά ο καταχωρητής του δεξιού παιδιού του κόμβου.

Για παράδειγμα, ο κόμβος 30 δίνει την εντολή:

```
add $11, $12, $11
```

Σε ειδικές περιπτώσεις, όπως για παράδειγμα σε εντολές πολλαπλασιασμού και διαίρεσης, η σύνταξη αυτή διαφοροποιείται, λόγω αναγκαστικής χρήσης των ειδικών καταχωρητών hi και lo, χωρίς όμως να επηρεάζεται η δέσμευση των καταχωρητών, αφού δεν απαιτούνται πρόσθετοι καταχωρητές για την παραγωγή των εντολών αυτών.

Όσο αφορά τα φύλλα του δέντρου, αυτά είναι είτε σταθερές, είτε αναγνωριστικά. Οι σταθερές μπορούν να φορτωθούν στον αντίστοιχο καταχωρητή, είτε με μία εντολή MIPS, εάν έχουν τιμή που να μπορεί να αναπαρασταθεί με 16 bits, ή με δύο εντολές, διαφορετικά. Στην τελευταία περίπτωση δεν επηρεάζεται η δέσμευση των καταχωρητών, για τον ίδιο λόγο όπως προηγουμένως.

Για παράδειγμα, ο κόμβος 9 δίνει την εντολή:

```
ori $11, $0, 0x28
```

Υποθέτουμε ότι οι μεταβλητές στοίβας διευθυνσιοδοτούνται με μετατόπιση σχετικά με την κορυφή της στοίβας, η τιμή της οποίας περιέχεται στον καταχωρητή $sp[6]. Μ' άλλα λόγια, η διεύθυνσή τους αναπαριστάται σα μια σταθερά, η οποία συνήθως μπορεί να ενσωματωθεί στην εντολή που βρίσκει την τιμή δεξιάς προσπέλασης της μεταβλητής.
Για παράδειγμα, ο κόμβος 1, όταν δίνει τιμή δεξιάς προσπέλασης, ώστε αυτή να χρησιμοποιηθεί στον κόμβο 3, παράγει την εντολή:

```
lw $10, 32($sp)
```

Οι μεταβλητές του χώρου στατικών δεδομένων, τέλος, μπορούν να διευθυνσιοδοτηθούν απ' ευθείας με τη διεύθυνση που έχουν στον αντίστοιχο πίνακα διευθύνσεων, με την προϋπόθεση ότι ο χώρος στατικών δεδομένων είναι ο πρώτος χώρος στο συνολικό χώρο δεδομένων του προγράμματος. Εναλλακτικά, μπορούμε να υποθέσουμε ότι ο χώρος στατικών δεδομένων διευθυνσιοδοτείται σχετικά με μια θέση μνήμης που περιέχεται σε ένα δεύτερο καταχωρητή, τον $gp[7]. Η επιλογή της μεθόδου διευθυνσιοδότησης που χρησιμοποιείται γίνεται από τον προγραμματιστή του μεταγλωττιστή, με την προφανή απαίτηση η τελική διεύθυνση να προκύπτει σωστή.
Χρησιμοποιώντας την πρώτη μέθοδο, η φόρτωση της διεύθυνσης μιας μεταβλητής θα γίνει με μία ή δύο εντολές MIPS, ανάλογα αν η σταθερά που αποτελεί τη διεύθυνση μπορεί ή όχι να αναπαρασταθεί με 16 bits. Η φόρτωση της τιμής της μεταβλητής θα γίνει με μία ακόμα εντολή.

Με βάση τα παραπάνω, ο τελικός κώδικας MIPS που αποτιμά την έκφραση που μας δόθηκε δίνεται πιο κάτω:

Κόμβος	Κώδικας	Σχόλια
1	`lw $10, 32($sp)`	Δεξιά προσπέλαση μεταβλητής j
2	`ori $11, $0, 2`	Σταθερά 2
3	`mult $10, $11` `mflo $10`	Πολλαπλασιασμός
4	`lui $11, 0x3d` `ori $11, $11, 0xa540`	Διεύθυνση 4040000 = 0x3da540 (hex)
5	`add $10, $11, $10`	Πρόσθεση
6	`lh $10, 0($10)`	Δεξιά προσπέλαση στοιχείου πίνακα m (2 bytes)
7	`lh $11, 40($sp)`	Δεξιά προσπέλαση μεταβλητής w (2 bytes)
8	`mult $11, $10` `mflo $10`	Πολλαπλασιασμός
9	`ori $11, $0, 0x28`	Σταθερά 40 = 0x28 (hex)
10	`mult $10, $11` `mflo $10`	Πολλαπλασιασμός
11	`lw $11, 28($sp)`	Δεξιά προσπέλαση μεταβλητής i
12	`ori $12, $0, 0x9c40`	Σταθερά 40000 = 0x9c40 (hex)
13	`mult $11, $12` `mflo $11`	Πολλαπλασιασμός
14	`ori $12, $0, 0x9c40`	Διεύθυνση 40000 = 0x9c40 (hex)
15	`add $11, $12, $11`	Πρόσθεση
16	`add $10, $11, $10`	Πρόσθεση
17	`lw $11, 36($sp)`	Δεξιά προσπέλαση μεταβλητής k
18	`addi $11, $11, 1`	Αύξηση κατά 1
19	`ori $12, $0, 4`	Σταθερά 4
20	`mult $11, $12` `mflo $11`	Πολλαπλασιασμός
21	`add $10, $10, $11`	Πρόσθεση
22	`lw $10, 0($10)`	Δεξιά προσπέλαση στοιχείου πίνακα b

[6] Ο καταχωρητής $sp είναι ένας από τους καταχωρητές σταθερής υποδιαστολής, συνήθως ο $29.
[7] Ο $gp είναι συνήθως ο καταχωρητής $28.

23	`lw $11, 28($sp)`	Δεξιά προσπέλαση μεταβλητής i
24	`lw $12, 32($sp)`	Δεξιά προσπέλαση μεταβλητής j
25	`mult $11, $12` `mflo $11`	Πολλαπλασιασμός
26	`addi $11, $11, 1`	Αύξηση κατά 1
27	`ori $12, $0, 0x28`	Σταθερά 40 = 0x28 (hex)
28	`mult $11, $12` `mflo $11`	Πολλαπλασιασμός
29	`ori $12, $0, 0`	Διεύθυνση 0
30	`add $11, $12, $11`	Πρόσθεση
31	`lw $12, 36($sp)`	Δεξιά προσπέλαση μεταβλητής k
32	`addi $12, $12, -2`	Μείωση κατά 2
33	`ori $13, $0, 4`	Σταθερά 4
34	`mult $12, $13` `mflo $12`	Πολλαπλασιασμός
35	`add $11, $11, $12`	Πρόσθεση
36	`lw $11, 0($11)`	Δεξιά προσπέλαση στοιχείου πίνακα a
37	`ori $12, $0, 5`	Σταθερά 5
38	`div $11, $12` `mflo $11`	Διαίρεση
39	`mult $11, $10` `mflo $10`	Πολλαπλασιασμός
40	`lw $11, 36($sp)`	Δεξιά προσπέλαση μεταβλητής j
41	`addi $11, $11, 3`	Αύξηση κατά 3

42	`lui $12, 0x3d`
	`ori $12, $12, 0xa608`
	`lw $12, 0($12)`
		Διεύθυνση 4040200 = 0x3da608 (hex) και δεξιά προσπέλαση μεταβλητής c

43	`mult $12, $11` `mflo $11`	Πολλαπλασιασμός
44	`sub $10, $10, $11`	Αφαίρεση

Άσκηση 9-4:

Θεωρήστε τον τελικό ενδιάμεσο κώδικα που προέκυψε μετά την εύρεση και απαλοιφή κοινών υποεκφράσεων στην άσκηση 8-1.
Α. Μετασχηματίστε τον κώδικα αυτόν στην απλούστερη δυνατή μορφή πριν την παραγωγή τελικού κώδικα, αρχικά εφαρμόζοντας ένα σχήμα παραγωγής κώδικα MIPS για τον κόμβο τύπου if_statement, και στη συνέχεια απαλείφοντας τον τελεστή αναφοράς σε στοιχείο πίνακα '[ ]', και εφαρμόζοντας υποβιβασμό ισχύος, όπου μπορείτε.
Β. Προχωρήστε σε συνολική δέσμευση καταχωρητών για τον κώδικα αυτόν με τη μέθοδο χρωματισμού του γράφου αλληλεπιδράσεων, ώστε να επιτύχετε τον ελάχιστο αριθμό καταχωρητών.
Γ. Αν οι διαθέσιμοι καταχωρητές είναι κατά ένας λιγότεροι από όσους προκύπτουν από την προηγούμενη δέσμευση, να την επαναλάβετε, χρησιμοποιώντας διάχυση στη μνήμη για έναν ή παραπάνω κόμβους του κώδικα, μέχρι να πετύχετε δέσμευση με όσους ακριβώς καταχωρητές διατίθενται.
Δ. Να δώσετε έναν τελικό κώδικα MIPS για την παραπάνω έκφραση, με βάση τη δέσμευση καταχωρητών που βρήκατε στο προηγούμενο ερώτημα, υποθέτοντας ότι οι απαιτούμενες μετατοπίσεις στο χώρο δεδομένων είναι: (G:1) → (G:0), (G:2) → (G:4000), (L:6) → (L:32) και (L:7) → (L:36) για μεταβλητές, και από (L:304) και πέρα για διάχυση καταχωρητών.

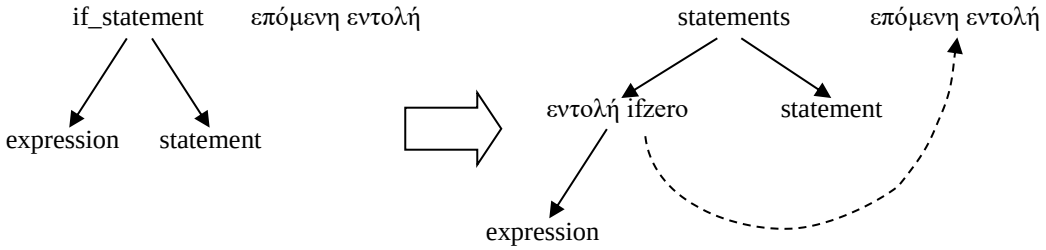
Απάντηση

Α. Η εντολή if που απεικονίζεται στον κόμβο τύπου *if_statement* είναι από τις εντολές που μεταφράζονται εύκολα σε τελικό κώδικα, μια που αρκεί γι' αυτό μια απλή εντολή διακλάδω-

σης μετά την αποτίμηση της έκφρασης συνθήκης. Έτσι, το σχήμα μετάφρασης που θα ακολουθήσουμε θα είναι το εξής:

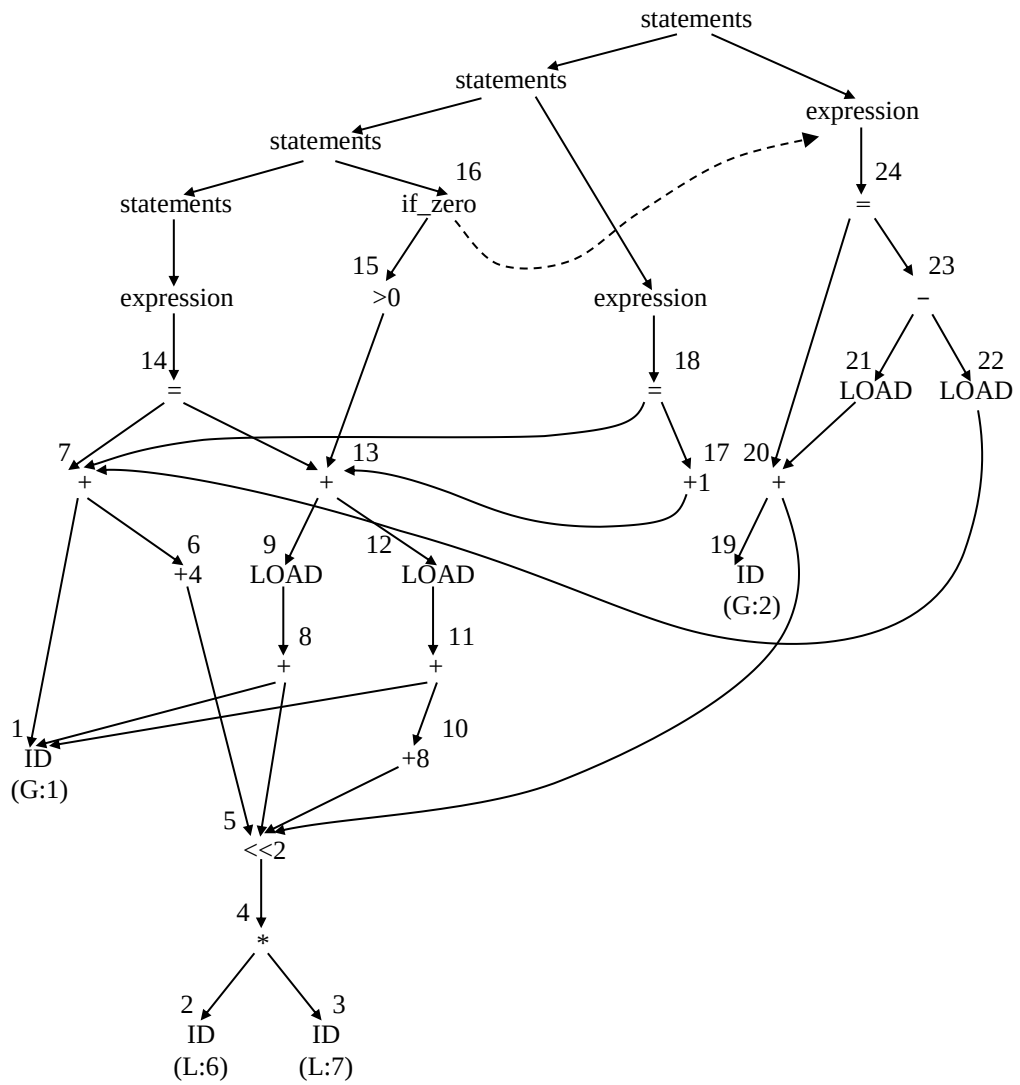
1. Κατασκευή ενός νέου κόμβου *εντολής ifzero* που εκτελεί άλμα αν το τελούμενό της έχει μηδενική τιμή. Ο κόμβος αυτός θα έχει δύο παιδιά, ένα δείκτη στο δέντρο της έκφρασης συνθήκης – δηλαδή το αριστερό παιδί – της αρχικής εντολής if, και ένα δείκτη στον κόμβο της εντολής που ακολουθεί την εντολή if.
2. Σύνδεση σε δυαδικό δέντρο του νέου κόμβου, με τον κόμβο της εντολής που αντιστοιχεί στην απλή ή στη σύνθετη εντολή που περιέχεται στην εντολή if – δηλαδή το δεξί παιδί του κόμβου if_statement.
3. Αντικατάσταση του αρχικού κόμβου της εντολής if με το δέντρο των δύο κόμβων που κατασκευάσαμε.

Το σχήμα αυτό απεικονίζεται στο ακόλουθο διάγραμμα:



Εφαρμόζουμε το παραπάνω σχήμα παραγωγής τελικού κώδικα στον κόμβο if_statement του ενδιάμεσου κώδικα της άσκησης 8-1, και από το νέο, απλούστερο ενδιάμεσο κώδικα που λαμβάνουμε συνεχίζουμε με την επίλυση αναφοράς για στοιχεία πίνακα. Εφόσον οι πίνακες που έχουμε είναι όλοι διανύσματα ακεραίων, η επίλυση αναφοράς για ένα στοιχείο γίνεται με πρόσθεση της αρχικής διεύθυνσης του πίνακα και της τιμής του δείκτη, πολλαπλασιασμένου επί 4 που είναι το μέγεθος ενός ακεραίου. Επειδή ο πολλαπλασιασμός επί 4 γίνεται πιο απλά με αριστερή ολίσθηση κατά 2 θέσεις, ο υποβιβασμός ισχύος επιβάλλει την αντικατάσταση ενός τέτοιου πολλαπλασιασμού με την ισοδύναμη αριστερή ολίσθηση. Έτσι, ο ενδιάμεσος κώδικας που έχουμε από την άσκηση 8-1 μετασχηματίζεται στην τελική μορφή που δίνεται στην επόμενη σελίδα, όπου η αριστερή ολίσθηση τοποθετήθηκε κάτω από την πρόσθεση των σταθερών 1 και 2, έτσι ώστε να πάρουμε την ολισθημένη τιμή σαν κοινή υποέκφραση, οπότε οι σταθερές πολλαπλασιάστηκαν επί 4.

Β. Για να προχωρήσουμε σε δέσμευση καταχωρητών με τη μέθοδο χρωματισμού του γράφου αλληλεπιδράσεων, κατ' αρχήν αριθμούμε τους κόμβους του τελικού ενδιάμεσου κώδικα από κάτω προς τα επάνω και από αριστερά προς τα δεξιά, όπως δείχνουμε στον κώδικα. Η αρίθμηση αυτή προκύπτει με διαπέραση κατά βάθος του συνολικού ενδιάμεσου κώδικα, όπου αγνοούμε όχι μόνο κόμβους που έχουμε ήδη αριθμήσει, αλλά και κόμβους που δεν παράγουν τελικό κώδικα, όπως κόμβους τύπου expression ή statements. Ο κόμβος τύπου ifzero παράγει μια εντολή διακλάδωσης, με τελούμενο εισόδου το τελούμενο εξόδου της εντολής του κόμβου ">0".
Για να κατασκευάσουμε το γράφο αλληλεπιδράσεων, θα πρέπει να προσέξουμε την επίδραση που έχει η απαλοιφή των κοινών υποεκφράσεων στη διάρκεια ζωής των κόμβων. Η διάρκεια ζωής ενός κόμβου ξεκινά με τον κώδικα του κόμβου και τερματίζεται με τη χρήση της τιμής του κόμβου από τον κώδικα του πατρικού κόμβου. Στην περίπτωση όμως που ο κόμβος αντιστοιχεί σε κοινή υποέκφραση, η απαλοιφή οδηγεί στη σύνδεση του κόμβου με πολλαπλούς πατρικούς κόμβους. Επομένως, η διάρκεια ζωής του κόμβου θα παραταθεί μέχρι τον κώδικα του πιο μακρινού πατέρα, εφόσον η τιμή του κόμβου θα πρέπει να παραμείνει ζωντανή στον κώδικα όλων των ενδιάμεσων κόμβων. Για παράδειγμα, στον παραπάνω ενδιάμεσο κώδικα, η διάρκεια ζωής του κόμβου 1 καλύπτει τους κόμβους από τον 1 μέχρι τον κόμβο 11. Ο πιο μακρινός πατρικός κόμβος είναι αυτός με το μεγαλύτερο αριθμό, από τη στιγμή που η αρίθμηση αποτελεί και τη σειρά εντολών τελικού κώδικα.

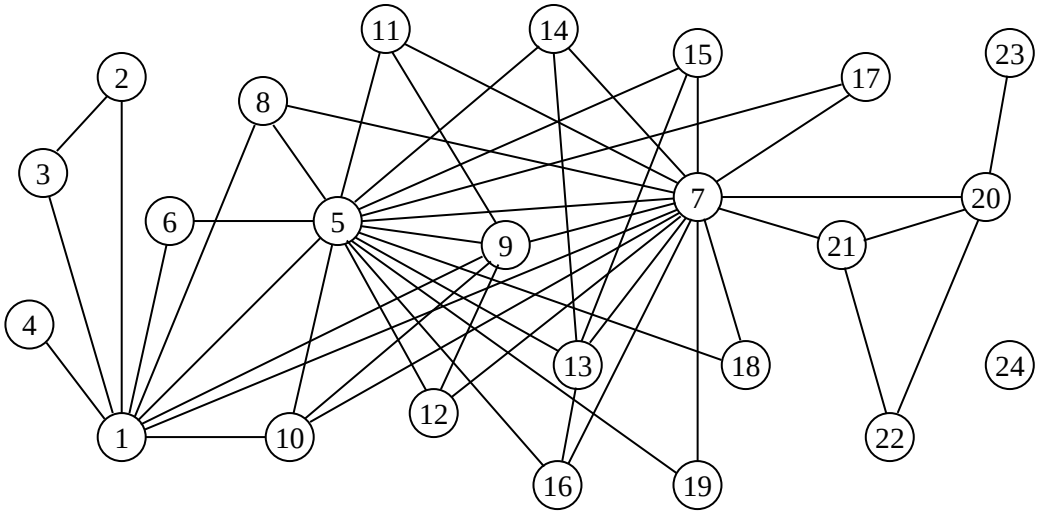


Γνωρίζοντας τις διάρκειες ζωής των κόμβων, μπορούμε να κατασκευάσουμε το γράφο αλληλεπιδράσεων, εξετάζοντας τους κόμβους με τη σειρά αρίθμησης, δεδομένου ότι κάθε νέος κόμβος αλληλεπιδρά με όλους τους κόμβους που ακολουθούν μέχρι τον πιο μακρινό πατέρα του[8]. Έτσι, προκύπτει ο γράφος της επόμενης σελίδας, ο οποίος παρατηρούμε ότι είναι πιο πολύπλοκος από εκείνους που είδαμε στην προηγούμενη άσκηση 9-3, κάτι που όμως είχαμε προβλέψει. Πιο συγκεκριμένα, ο μέγιστος πλήρης υπογράφος του γράφου αυτού είναι ο υπογράφος που σχηματίζεται από τους κόμβους 1, 5, 7, 9 και 10, ο οποίος δε μπορεί να χρωματιστεί με λιγότερα από 5 χρώματα, όσες είναι οι κορυφές του. Άρα ο χρωματικός αριθμός του γράφου που κατασκευάσαμε είναι τουλάχιστον 5.

Η ταξινόμηση των κόμβων του γράφου σε φθίνουσα σειρά βαθμού δίνει το εξής αποτέλεσμα:
7, 5, 1, 9, 13, 10, 20, 8, 11, 12, 14, 15, 16, 21, 2, 3, 6, 17, 18, 19, 22, 4, 23, 24.

Ο χρωματισμός του γράφου αλληλεπιδράσεων με την παραπάνω σειρά χρωματισμού κόμβων, έτσι ώστε δύο κόμβοι που αλληλεπιδρούν να μην έχουν το ίδιο χρώμα, και με διαθέσιμα

[8] Στην περίπτωση αυτή η διαδικασία είναι σχετικά απλή. Αν όμως ο κώδικάς μας ήταν το πλήρες σώμα κάποιου βρόχου, τότε μια συνολική δέσμευση καταχωρητών θα μπορούσε να είναι πιο δύσκολη. Αυτό θα συνέβαινε, αν θέλαμε να ελαχιστοποιήσουμε τις προσπελάσεις μνήμης, διατηρώντας τιμές σε καταχωρητές, επειδή θα έπρεπε να εντοπίσουμε και διάρκειες ζωής που αλλάζουν από επανάληψη σε επανάληψη του βρόχου. Διαφορετικά, θα αντιμετωπίζαμε κάθε επανάληψη σαν ανεξάρτητο κώδικα, ο οποίος θα έπρεπε να φορτώνει τιμές από τη μνήμη και στο τέλος να τις αποθηκεύει.



χρώματα τους καταχωρητές $8 έως $12, δίνει το αποτέλεσμα που καταγράφεται στον πίνακα που ακολουθεί:

Καταχωρητής	Κόμβοι
$8	7, 2, 6, 22, 23
$9	5, 20, 3, 4, 24
$10	1, 13, 11, 12, 21
$11	9, 8, 14, 15, 16
$12	10, 17, 18, 19

Παρατηρούμε ότι ο χρωματισμός αυτός αντιστοιχεί στο κάτω φράγμα του χρωματικού αριθμού που είχαμε υπολογίσει νωρίτερα, κι επομένως είναι βέλτιστος.

Γ. Αν τώρα δεν διαθέτουμε πέντε καταχωρητές, αλλά τέσσερις, τότε θα πρέπει να βρούμε κάποιον τρόπο να ανακατασκευάσουμε το γράφο αλληλεπιδράσεων, ώστε να προκύψει ένας νέος γράφος με μικρότερο χρωματικό αριθμό. Η τεχνική που θα μελετήσουμε εδώ είναι η διάχυση, δηλαδή η χρήση μιας προσωρινής μεταβλητής στοίβας για αποθήκευση και επαναφόρτωση κάποιας τιμής από αυτές που παράγονται από τον κώδικα. Η αποθήκευση της τιμής θα γίνεται αμέσως μετά την παραγωγή της ή κάποια χρήση της, και η επακόλουθη φόρτωση θα γίνεται αμέσως πριν από κάποια επόμενη χρήση της. Όσο η τιμή είναι αποθηκευμένη στη στοίβα, δεν υπάρχει αλληλεπίδραση με κανένα κόμβο του ενδιάμεσου κώδικα, κι επομένως όλες οι αλληλεπιδράσεις του κόμβου που παράγει την τιμή με τους κόμβους μεταξύ της αποθήκευσης και της φόρτωσης μπορούν να διαγραφούν από το γράφο αλληλεπιδράσεων. Ο γράφος έτσι γίνεται λιγότερο πολύπλοκος, και ίσως μπορεί να επαναχρωματιστεί με λιγότερα χρώματα. Αν όχι, τότε θα πρέπει να επαναλάβουμε την ίδια διαδικασία όσες φορές χρειάζεται, κάθε φορά επιλέγοντας μία ακόμα τιμή του κώδικα για προσωρινή αποθήκευση, μέχρι επιτέλους να πετύχουμε τον επιθυμητό χρωματικό αριθμό.

Αυτό που είναι το κύριο ζητούμενο στην τεχνική αυτή είναι μια καλή ευριστική μέθοδος, ώστε να επιλέξουμε για διάχυση τον καταλληλότερο κόμβο, και να έχουμε τις καλύτερες πιθανότητες ο νέος γράφος αλληλεπιδράσεων να χρωματίζεται με λιγότερα χρώματα με την πρώτη προσπάθεια. Θα αναπτύξουμε τη μέθοδο αυτή στηριζόμενοι στις εξής παρατηρήσεις:

1. Επειδή ο χρωματικός αριθμός περιορίζεται από το μεγαλύτερο πλήρη υπογράφο του γράφου αλληλεπιδράσεων, καλό θα είναι ο κόμβος που θα επιλέξουμε να ανήκει σε αυτό τον υπογράφο. Ειδικότερα, αν ο χρωματικός αριθμός ταυτίζεται με το μέγεθος του υπογράφου, τότε η επιλογή θα γίνεται υποχρεωτικά μέσα από αυτόν. Αν υπάρχουν πολλοί τέτοιοι υπογράφοι με τον ίδιο αριθμό κόμβων, θα επιλέξουμε κάποιον κοινό κόμβο, αλλιώς θα πρέπει να επιλέξουμε εξαρχής πολλαπλούς κόμβους, έναν από κάθε υπογράφο.

2. Για να απλουστεύσουμε το γράφο αλληλεπιδράσεων όσο γίνεται περισσότερο, θα πρέπει να επιλέξουμε μεταξύ κόμβων μεγάλου βαθμού, ώστε να απαλείψουμε το μέγιστο δυνατό αριθμό αλληλεπιδράσεων.
3. Αν ο υποψήφιος κόμβος έχει πολλούς πατέρες, τότε για να αποφύγουμε πολλαπλές φορτώσεις από τη μνήμη, μπορούμε να επιλέξουμε για διάχυση μόνο εκείνο το διάστημα ζωής του κόμβου από την παραγωγή της τιμής του μέχρι την πρώτη χρήση της ή από μια χρήση της τιμής του μέχρι την επόμενη, στο οποίο εμφανίζονται οι περισσότερες αλληλεπιδράσεις. Βέβαια, αν ο χρωματικός αριθμός του γράφου αλληλεπιδράσεων ταυτίζεται με το μέγεθος του μέγιστου πλήρους υπογράφου του, τότε τα διαστήματα που πρέπει να θεωρήσουμε είναι εκείνα που περιλαμβάνουν κόμβους από τον πλήρη υπογράφο του πρώτου κριτηρίου, αλλιώς δε θα καταφέρουμε να μειώσουμε το μέγεθός του.

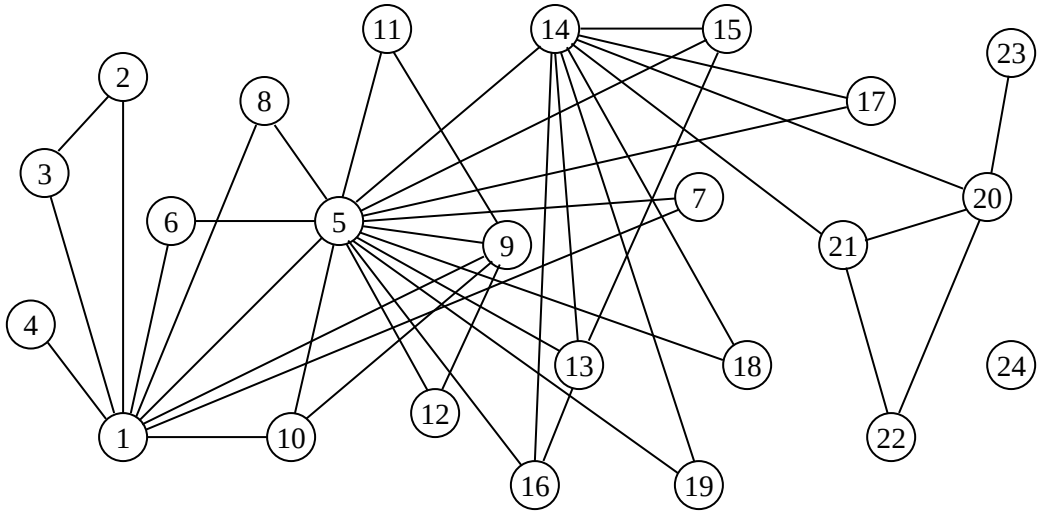
Έτσι λοιπόν, μπορούμε να επιλέγουμε έναν υποψήφιο κόμβο με βασικό κριτήριο να βρίσκεται σε μέγιστο πλήρη υπογράφο του γράφου αλληλεπιδράσεων. Από εκεί και πέρα, θα πρέπει να εκτιμούμε ένα συνδυασμό των υπόλοιπων κριτηρίων, μια που δεν είναι ξεκάθαρο αν το δεύτερο ή το τρίτο κριτήριο θα πρέπει να έχουν προτεραιότητα στην επιλογή. Το τρίτο κριτήριο θα παίζει τον καθοριστικό ρόλο, ιδιαίτερα αν οι βέλτιστοι κόμβοι του δεύτερου κριτηρίου έχουν παραπλήσιο βαθμό.
Στο συγκεκριμένο κώδικα που έχουμε, ο πλήρης υπογράφος που μας απασχολεί είναι αυτός που σχηματίζεται από τους κόμβους 1, 5, 7, 9 και 10. Από αυτούς τους κόμβους, οι κόμβοι 7 και 5 έχουν εμφανώς μεγαλύτερο βαθμό από τους υπόλοιπους, 16 και 15 αντίστοιχα, έναντι 9 του 1, 6 του 9 και 4 του 10. Όμως και οι δύο έχουν παραπλήσιο βαθμό, οπότε δεν είναι βέβαιο ότι ο 7 αποτελεί καλύτερο υποψήφιο για διάχυση από τον 5. Η τιμή του κόμβου 7 χρησιμοποιείται από τους κόμβους 14, 18 και 22, ενώ η τιμή του κόμβου 5 χρησιμοποιείται από τους κόμβους 6, 8, 10 και 20. Παρατηρούμε ότι το μέγιστο διάστημα ζωής για προσωρινή αποθήκευση είναι το διάστημα από τον 11 έως τον 20 για τον κόμβο 5, όμως το διάστημα αυτό δεν περιλαμβάνει κανέναν από τους κόμβους 1, 7, 9 και 10. Έτσι, θα πρέπει να επιλέξουμε τον κόμβο 7 που έχει το αμέσως μικρότερο διάστημα, από τον κόμβο 8 έως τον 14.

Παρόλο που η αποθήκευση μιας τιμής γίνεται χωρίς κόστος χρώματος, η επαναφόρτωσή της απαιτεί κάποιον καταχωρητή. Άρα, υπάρχει κίνδυνος, το χρώμα που θα κερδίσουμε από την απαλοιφή των αλληλεπιδράσεων να το χάσουμε για τη φόρτωση! Ευτυχώς, επειδή η φόρτωση της τιμής γίνεται αμέσως πριν τη χρήση της, αυτή δημιουργεί μόνο μία πρόσθετη αλληλεπίδραση, αυτή με το άλλο παιδί του κόμβου χρήσης. Στην ουσία, μπορούμε να δούμε τη φόρτωση σαν έναν πρόσθετο κόμβο στον ενδιάμεσο κώδικα, που να τοποθετείται αμέσως κάτω από τον κόμβο χρήσης όπου τερματίζεται το διάστημα στο οποίο εφαρμόζεται η διάχυση, και μάλιστα προς την κατεύθυνση του κόμβου παραγωγής της τιμής, και ο οποίος να αριθμείται αμέσως πριν τον κόμβο χρήσης. Επειδή αυτός ο κόμβος αριθμείται αμέσως πριν τον πατέρα του, δε θα έχει αλληλεπίδραση με αυτόν και μπορεί να πάρει το χρώμα του, οπότε δε χρειάζεται καν να συμμετάσχει στο χρωματισμό του γράφου αλληλεπιδράσεων. Αντίθετα με αυτόν, όμως, όλα τα άλλα αδέλφια του θα πρέπει τώρα να έχουν αλληλεπίδραση με τον πατέρα τους, ώστε να εξασφαλιστεί ότι το χρώμα του πατέρα θα χρησιμοποιηθεί από αυτόν τον πρόσθετο κόμβο, και όχι από άλλο παιδί.
Στην περίπτωσή μας, με την επιλογή που κάναμε παραπάνω, θα πρέπει να επιβάλλουμε αλληλεπίδραση του κόμβου 13 με τον κόμβο 14, ώστε η φόρτωση της τιμής που παράγει ο κόμβος 7 να γίνεται στον ίδιο καταχωρητή με τον οποίο χρωματίζεται ο κόμβος 14. (Εδώ βέβαια αυτή η αλληλεπίδραση προϋπήρχε για άλλο λόγο, οπότε δεν προστίθεται τώρα.)

Με όλα τα παραπάνω, ο γράφος αλληλεπιδράσεων του κώδικά μας τροποποιείται ως εξής:

1. Αφαιρούμε τις αλληλεπιδράσεις μεταξύ του κόμβου 7 και των κόμβων 8 έως 14, αφού η τιμή που παράγει ο κόμβος 7 δε διατηρείται πια σε καταχωρητή.
2. Αντικαθιστούμε τις αλληλεπιδράσεις μεταξύ του κόμβου 7 και των κόμβων 15 έως 21 με αντίστοιχες αλληλεπιδράσεις με τον κόμβο 14, αφού η τιμή που παράγει ο κόμβος 7 φορτώνεται από τη μνήμη αμέσως πριν τον κόμβο 14 και με το χρώμα του τελευταίου.
3. Προσθέτουμε την αλληλεπίδραση μεταξύ των κόμβων 13 και 14, αν δεν προϋπάρχει.

Ο γράφος διαμορφώνεται έτσι ως εξής:



και ο χρωματισμός του θα έχει το ακόλουθο αποτέλεσμα:

Καταχωρητής	Κόμβοι
$8	5, 20, 2, 4, 24
$9	1, 14, 11, 12, 22, 23
$10	9, 13, 21, 3, 6, 7, 8
$11	10, 15, 16, 17, 18, 19

Παρατηρούμε ότι ο χρωματισμός δίνει ένα χρώμα λιγότερο από όσα λάβαμε στον προηγούμενο γράφο αλληλεπιδράσεων. Ο αριθμός χρωμάτων αντιστοιχεί στο χρωματικό αριθμό του γράφου, δεδομένου του πλήρους υπογράφου τεσσάρων κόμβων που σχηματίζουν οι κόμβοι 1, 5, 9 και 10. Ο υπογράφος αυτός προκύπτει αν από τον πλήρη υπογράφο πέντε κόμβων που είχαμε βρει στον προηγούμενο γράφο αλληλεπιδράσεων αφαιρέσουμε τον κόμβο 7, αυτό ακριβώς που πετύχαμε με τη διάχυση της τιμής του κόμβου 7 στη μνήμη.

Δ. Η παραγωγή τελικού κώδικα γίνεται με τη σειρά αρίθμησης των κόμβων του τελικού ενδιάμεσου κώδικα, λαμβάνοντας υπόψη τη διάχυση της τιμής του κόμβου 7, η οποία συμβαίνει για το διάστημα που ξεκινάει αμέσως μετά τον κόμβο 7 και τελειώνει αμέσως πριν τον κόμβο 14. Αν η θέση στη στοίβα που χρησιμοποιείται για τη διάχυση έχει μετατόπιση 304 στο εγγράφημα δραστηριοποίησης του τρέχοντος περιβάλλοντος, το μέγεθος καταχωρητών είναι 32 bits και η προσπέλαση των χώρων δεδομένων στη μνήμη γίνεται με τη βοήθεια των καταχωρητών $gp και $sp, για τον καθολικό και τον τοπικό χώρο δεδομένων αντίστοιχα, τότε ο κώδικας που προκύπτει θα είναι ο ακόλουθος:

Κόμβος	Κώδικας	Σχόλια
1	`addiu $9, $gp, 0`	Αριστερή προσπέλαση πίνακα
2	`lw $8, 32($sp)`	Δεξιά προσπέλαση βαθμωτής μεταβλητής
3	`lw $10, 36($sp)`	Δεξιά προσπέλαση βαθμωτής μεταβλητής
4	`mult $8, $10` `mflo $8`	Πολλαπλασιασμός
5	`sll $8, $8, 2`	Αριστερή ολίσθηση
6	`addiu $10, $8, 4`	Πρόσθεση της σταθεράς 4
7	`addu $10, $9, $10` `sw $10, 304($sp)`	Πρόσθεση και διάχυση του αποτελέσματος
8	`addu $10, $9, $8`	Πρόσθεση
9	`lw $10, 0($10)`	Δεξιά προσπέλαση στοιχείου πίνακα
10	`addiu $11, $8, 8`	Πρόσθεση της σταθεράς 8

11	`addu $9, $9, $11`	Πρόσθεση
12	`lw $9, 0($9)`	Δεξιά προσπέλαση στοιχείου πίνακα
13	`add $10, $10, $9`	Πρόσθεση
14	`lw $9, 304($sp)` `sw $10, 0($9)`	Επαναφόρτωση και ανάθεση
15	`slti $11, $0, $10`	Σύγκριση
16	`beq $11,$0, Nxt`	Διακλάδωση
17	`addi $11, $10, 1`	Αύξηση κατά 1
18	`sw $11, 0($9)`	Ανάθεση
19	`Nxt: addiu $11, $gp, 4000`	Αριστερή προσπέλαση πίνακα
20	`addu $8, $11, $8`	Πρόσθεση
21	`lw $10, 0($8)`	Δεξιά προσπέλαση στοιχείου πίνακα
22	`lw $9, 0($9)`	Δεξιά προσπέλαση στοιχείου πίνακα
23	`sub $9, $10, $9`	Αφαίρεση
24	`sw $9, 0($8)`	Ανάθεση

Οι κόμβοι 18 και 22 που στον ενδιάμεσο κώδικα παρουσιάζονται να έχουν τελούμενο εισόδου από τον κόμβο 7, τώρα έχουν τελούμενο εισόδου από τον κόμβο 14, ενώ ο κόμβος 14 αντί να έχει τελούμενο εισόδου από τον κόμβο 7, τώρα έχει τελούμενο εισόδου που επαναφορτώνεται από τη μνήμη με την εντολή φόρτωσης που προηγείται.
Αξίζει τέλος να παρατηρήσουμε την υλοποίηση της εντολής if μέσω της διακλάδωσης με προορισμό την εντολή με ετικέτα Nxt. Η συνθήκη αποτιμάται στον καταχωρητή $11, ο οποίος και ελέγχεται από την εντολή διακλάδωσης, ώστε να εκτελεστεί ή όχι ο κώδικας των κόμβων 17 και 18 που προηγείται του προορισμού. Λόγω της εντολής if, η δεξιά προσπέλαση του στοιχείου πίνακα στον κόμβο 22 γίνεται με φόρτωση από τη διεύθυνση που παρέχει ο κόμβος 14 – μετά τη διάχυση, και όχι απ' ευθείας από την τιμή του κόμβου 17, κάτι που θα κάναμε μέσω διάδοσης αντιγράφων αν δεν υπήρχε διακλάδωση, δεδομένου ότι ο κόμβος 17 παρέχει την τελευταία τιμή γι' αυτό το στοιχείο πίνακα. Από τη στιγμή που υπάρχει διακλάδωση, το στοιχείο πίνακα μπορεί και να μην πάρει την τιμή του κόμβου 17, κι έτσι είμαστε υποχρεωμένοι να φορτώσουμε την τιμή του από τη μνήμη, η οποία θα περιέχει τη σωστή τιμή, αφού αυτή αποθηκεύεται είτε από τον κόμβο 18 αν εκτελεστεί ο κώδικας των κόμβων 17 και 18 είτε από τον κόμβο 14 αν δεν εκτελεστεί.

Άσκηση 9-5:

Θεωρήστε μια γλώσσα προγραμματισμού που υποστηρίζει φωλιασμένες δηλώσεις υποπρογραμμάτων με στατικό δέσιμο, και πέρασμα παραμέτρων κατ' αξία, κατ' αναφορά και κατ' όνομα. Έστω το παρακάτω πρόγραμμα P αυτής της γλώσσας:

```
program P:
      int a = 3, b = -1;
      int function M (int x; var int y; name int z):
            int a = x+z;
            int function N (int i; name int j):
                  y = a+j;
                  if i >= 0 return j * N(i-1, i+j);
                  else return j;
            end N;
            b = b*z;
            return N(a, x+y+z);
      end M;
      print M(b, a, a+b);
end P.
```

Λαμβάνοντας υπ' όψη ότι (α) η συνάρτηση N είναι φωλιασμένη μέσα στη M, (β) οι μεταβλητές του P είναι στατικές, ενώ των M και N είναι μεταβλητές στοίβας, (γ) το πέρασμα των παραμέ-

τρων γίνεται κατ' αναφορά, αν μιας δήλωσης παραμέτρου προηγείται η λέξη-κλειδί "var", κατ' όνομα, αν μιας δήλωσης παραμέτρου προηγείται η λέξη-κλειδί "name", και κατ' αξία σε άλλη περίπτωση, και (δ) οι εκφράσεις αποτιμώνται από αριστερά προς τα δεξιά, βρείτε την τιμή που τυπώνει το πρόγραμμα με την εντολή print, *αναλύοντας την εκτέλεσή του ως εξής:*
Α. Δώστε τα περιεχόμενα του χώρου δεδομένων της μνήμης με την είσοδο στον κώδικα του P, με την είσοδο και έξοδο σε κάθε κλήση των συναρτήσεων M και N, αλλά και με την έξοδο από το P. Υποθέστε ότι κάθε εγγράφημα δραστηριοποίησης περιλαμβάνει σύνδεσμο προσπέλασης, διεύθυνση επανόδου, τιμή αποτελέσματος, παραμέτρους και τοπικές μεταβλητές, με τη σειρά που αναφέρθηκαν. Θεωρήστε ότι η στοίβα αναπτύσσεται προς τα κάτω, αμέσως μετά το χώρο στατικών δεδομένων, με τιμή του δείκτη στοίβας πριν την εκκίνηση του P ίση με 1000, και με όλες τις θέσεις που μας απασχολούν να έχουν μέγεθος 4 bytes. Αγνοήστε τις τιμές για τη διεύθυνση επανόδου.
Β. Παράλληλα με την απάντηση στο προηγούμενο ερώτημα, εξηγήστε πότε, πώς και με τι αποτέλεσμα γίνεται η αποτίμηση κάθε παραμέτρου που περνάει κατ' όνομα. Υπενθυμίζεται ότι η αποτίμηση μιας τέτοιας παραμέτρου γίνεται στο καλούν περιβάλλον.

Απάντηση

Η άσκηση αυτή μας ζητάει να μελετήσουμε την εξέλιξη του χώρου δεδομένων της μνήμης για πρόγραμμα που δίνεται σε κάποια αρχική γλώσσα προγραμματισμού. Ένας τρόπος για να απαντήσουμε, είναι να μεταγλωττίσουμε το πρόγραμμα σε κάποια τελική γλώσσα και μετά να εκτελέσουμε τον αντίστοιχο κώδικα, εντολή προς εντολή, παρατηρώντας τις αλλαγές στη μνήμη, είτε στο χώρο στατικών δεδομένων είτε στη στοίβα.
Όμως εδώ μας ενδιαφέρει μόνο η λειτουργική συμπεριφορά του προγράμματος στη μνήμη, δηλαδή η διαχείριση του χώρου δεδομένων του και οι τιμές που γράφονται σε αυτόν κατά την εκτέλεση του προγράμματος. Έτσι, μπορούμε να αποφύγουμε τη μεταγλώττιση και να μελετήσουμε αυτή τη συμπεριφορά, προσομοιώνοντας την εκτέλεση του προγράμματος στο επίπεδο του αρχικού κώδικα, και αναπαριστώντας το χώρο δεδομένων της μνήμης σαν ένα πίνακα, με κάθε θέση του οποίου να αντιστοιχείται σε μια μεταβλητή του προγράμματος.

Εφ' όσον ο χώρος στατικών δεδομένων είναι αμέσως πάνω από τη στοίβα, κι επειδή το πρόγραμμα P διαθέτει δύο μεταβλητές, τις a και b, ο χώρος διευθύνσεων που αυτός καλύπτει είναι ο [1000..1007]. Αν οι μεταβλητές τοποθετούνται στο χώρο στατικών δεδομένων από κάτω προς τα πάνω και αρχικοποιούνται πριν την έναρξη της εκτέλεσης του P, ο χώρος δεδομένων θα έχει την ακόλουθη μορφή με την είσοδο στον κώδικα του P:

b:	-1	
a:	3	(1000)

όπου η διεύθυνση δίνεται σε παρένθεση στα δεξιά του περιεχομένου. Με διπλή γραμμή συμβολίσαμε την κορυφή – προς τα κάτω – του χώρου στατικών δεδομένων, κάτω από την οποία αρχίζει η στοίβα.

Το πρόγραμμα P δεν κάνει τίποτε άλλο από την εκτύπωση κάποιας τιμής, την οποία υπολογίζει με την κλήση της συνάρτησης M. Το εγγράφημα δραστηριοποίησης (ΕΔ) της M θα περιέχει μετά τις τρεις πρώτες θέσεις για τις τιμές του συνδέσμου προσπέλασης (ΣΠ), της διεύθυνσης επανόδου (ΔΕ) και τιμής αποτελέσματος (ΤΑ), τις τρεις παραμέτρους της, και αμέσως μετά την τοπική μεταβλητή a. Για την κλήση της M, το πρόγραμμα P υπολογίζει κατ' αρχήν τις πραγματικές παραμέτρους της και τις τοποθετεί κάτω από το δείκτη στοίβας, στις προκαθορισμένες θέσεις. Η πρώτη παράμετρος περνά κατ' αξία, κι επομένως το P αποτιμά την έκφραση "b" ως έκφραση δεξιάς προσπέλασης. Η δεύτερη παράμετρος περνά κατ' αναφορά, κι επομένως το P αποτιμά την έκφραση "a" ως έκφραση αριστερής προσπέλασης. Η τρίτη παράμετρος περνά κατ' όνομα, κι επομένως το πρόγραμμα περνά αντί κάποιας τιμής, ένα δείκτη

σε κατάλληλο κώδικα που αποτιμά την έκφραση “a+b” στο περιβάλλον του P κάθε φορά που η παράμετρος συναντάται στη συνάρτηση. Οι τρεις τιμές που προκύπτουν, δηλαδή η -1, η διεύθυνση της a, και ο δείκτης στον κώδικα αποτίμησης του “a+b”, τοποθετούνται στη στοίβα. Αμέσως πριν την κλήση, τοποθετείται στη στοίβα και η τιμή του ΣΠ, που λόγω στατικού δεσίματος, θα είναι η διεύθυνση 1000 όπου αρχίζει ο χώρος δεδομένων του P, δηλαδή ο χώρος στατικών δεδομένων. Το ΕΔ της συνάρτησης M ενεργοποιείται με την είσοδο σε αυτή, με μείωση του δείκτη στοίβας κατά 28, εφ’ όσον περιλαμβάνει 7 στοιχεία των 4 bytes. Έτσι, ο συνολικός χώρος δεδομένων θα έχει τη στιγμή αυτή την ακόλουθη μορφή:

b:	-1	
a:	3	(1000)
a:		
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)

όπου με “&P::a” συμβολίσαμε τη διεύθυνση της μεταβλητής a της μονάδας P. Όπως και με το χώρο στατικών δεδομένων, η αντιστοίχηση των θέσεων γίνεται από κάτω προς τα πάνω, με το ΕΔ της M να καλύπτει το χώρο διευθύνσεων [972..999]. Με τη νέα διπλή γραμμή συμβολίσαμε την κορυφή του ΕΔ, δηλαδή την τρέχουσα κορυφή της στοίβας.

Η πρώτη εντολή που εκτελείται στη συνάρτηση M είναι η ανάθεση “a = x+z”, που προέρχεται από την αρχικοποίηση της τοπικής μεταβλητής a. Η αποτίμηση της μεταβλητής x, που είναι παράμετρος κατ’ αξία, γίνεται με ανάγνωση από τη στοίβα της τιμής -1. Η αποτίμηση της z, που είναι παράμετρος κατ’ όνομα, γίνεται με εκτέλεση του κώδικα “a+b” στο περιβάλλον του P, και δίνει την τιμή 3-1=2. Η ανάθεση αποθηκεύει την τιμή -1+2=1 στην a.
Η επόμενη εντολή της M είναι η ανάθεση “b = b*z”. Η b είναι μη τοπική μεταβλητή και αναζητείται στο χώρο δεδομένων, στον οποίο δείχνει ο ΣΠ του ΕΔ της M, που εδώ αντιστοιχεί στο χώρο στατικών δεδομένων. Η b όντως ανήκει σε αυτόν τον χώρο, και η τιμή της που διαβάζεται από τη μνήμη είναι -1. Η αποτίμηση της z γίνεται με νέα εκτέλεση του κώδικα “a+b” στο περιβάλλον του P, και δίνει πάλι την τιμή 2. Η ανάθεση αποθηκεύει την τιμή -1*2=-2 στη μη τοπική μεταβλητή b του χώρου στατικών δεδομένων.
Η κλήση της M ολοκληρώνεται με εντολή επιστροφής στο πρόγραμμα P. Πριν την επιστροφή όμως πρέπει να αποτιμηθεί η έκφραση της ΤΑ, και γι’ αυτό πρέπει να κληθεί η συνάρτηση N. Το ΕΔ της N θα περιέχει τις θέσεις για τις τιμές των ΣΠ, ΔΕ και ΤΑ, και τις δύο παραμέτρους της N. Για την κλήση της N, η συνάρτηση M υπολογίζει τις πραγματικές παραμέτρους και τις τοποθετεί κάτω από το δείκτη στοίβας. Η πρώτη παράμετρος περνά κατ’ αξία, κι επομένως η έκφραση “a” αποτιμάται ως έκφραση δεξιάς προσπέλασης. Η δεύτερη παράμετρος περνά κατ’ όνομα, κι επομένως η M περνά ένα δείκτη σε κατάλληλο κώδικα που αποτιμά την έκφραση “x+y+z” στο περιβάλλον της M. Οι τιμές που προκύπτουν, δηλαδή η 1 και ο δείκτης στον κώδικα “x+y+z”, τοποθετούνται στη στοίβα. Αμέσως πριν την κλήση, η M τοποθετεί στη στοίβα και την τιμή του ΣΠ, που θα είναι η διεύθυνση 972 όπου αρχίζει ο χώρος δεδομένων της M, εφ’ όσον η N είναι φωλιασμένη μέσα στη συνάρτηση M. Το ΕΔ της N ενεργοποιείται με την είσοδο σε αυτή, με μείωση του δείκτη στοίβας κατά 20, εφ’ όσον περιλαμβάνει 5 στοιχεία των 4 bytes, και καλύπτει το χώρο διευθύνσεων [952..971]. Έτσι, ο συνολικός χώρος δεδομένων θα έχει τη στιγμή αυτή τη μορφή:

b:	-2	
a:	3	(1000)
a:	1	

z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	
i:	1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(952)

Η πρώτη εντολή που εκτελείται στη συνάρτηση Ν είναι η ανάθεση “y = a+j”. Η a είναι μη τοπική μεταβλητή και αναζητείται στο χώρο δεδομένων, στον οποίο δείχνει ο ΣΠ του ΕΔ της Ν, που εδώ αντιστοιχεί στο ΕΔ της Μ. Η a βρίσκεται σε αυτόν τον χώρο, και η τιμή της που διαβάζεται από τη στοίβα είναι 1. Η αποτίμηση της j, που είναι παράμετρος κατ’ όνομα, γίνεται με εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της Μ. Στο περιβάλλον αυτό, η x είναι παράμετρος κατ’ αξία με τιμή -1, η y είναι παράμετρος κατ’ αναφορά με τιμή την τιμή της μεταβλητής a του Ρ, δηλαδή 3, ενώ η z είναι παράμετρος κατ’ όνομα, η οποία πρέπει να αποτιμηθεί με εκτέλεση του κώδικα που αποτιμά την έκφραση “a+b” στο περιβάλλον του προγράμματος Ρ, δίνοντας την τιμή 3-2=1. Η τιμή που προκύπτει για την j είναι η -1+3+1=3. Η y είναι μη τοπική μεταβλητή στη συνάρτηση Ν, και βρίσκεται στο ΕΔ της Μ. Επειδή όμως στο περιβάλλον της Μ είναι παράμετρος κατ’ αναφορά, η ανάθεση της τιμής 1+3=4 γίνεται στην πραγματικότητα στη μεταβλητή a του Ρ.
Στη συνέχεια εκτελείται η εντολή “if”. Η έκφραση “i>=0”, λόγω της τιμής της i, αποτιμάται ως “Αληθής”. Έτσι, εκτελείται η πρώτη από τις δύο εντολές επιστροφής στη συνάρτηση Μ. Για τον υπολογισμό της ΤΑ, απαιτείται η αποτίμηση μιας έκφρασης γινομένου, με πρώτο όρο την j και δεύτερο όρο το αποτέλεσμα αναδρομικής κλήσης της Ν. Η αποτίμηση της j γίνεται με νέα εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της Μ, ο οποίος με τη σειρά του εκτελεί τον κώδικα “a+b” στο περιβάλλον του Ρ, με τελική τιμή για την j την -1+4+(4-2)=5.
Για την αναδρομική κλήση της Ν, οι πραγματικές παράμετροι υπολογίζονται από την τρέχουσα κλήση της Ν και τοποθετούνται κάτω από το δείκτη στοίβας. Η πρώτη παράμετρος περνά κατ’ αξία, με τιμή που προκύπτει από την αποτίμηση της έκφρασης “i-1”. Η δεύτερη παράμετρος περνά κατ’ όνομα, με ένα δείκτη σε κατάλληλο κώδικα που αποτιμά την έκφραση “i+j” στο τρέχον περιβάλλον της Ν. Οι τιμές που προκύπτουν, δηλαδή η 0 και ο δείκτης στον κώδικα “i+j”, τοποθετούνται στη στοίβα. Αμέσως πριν την κλήση, τοποθετείται στη στοίβα και η τιμή του ΣΠ, που θα είναι πάλι η διεύθυνση 972, μια που ο ΣΠ της Ν δεν αλλάζει μέσα από αναδρομικές κλήσεις. Ένα νέο ΕΔ της συνάρτησης Ν ενεργοποιείται με την αναδρομική είσοδο σε αυτή, με περαιτέρω μείωση του δείκτη στοίβας κατά 20. Το νέο ΕΔ καλύπτει έτσι το χώρο διευθύνσεων [932..951]. Ο συνολικός χώρος δεδομένων θα έχει τη στιγμή αυτή τη μορφή:

b:	-2	
a:	4	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	

i:	1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(952)
j:	“i+j”	
i:	0	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(932)

Η ανάθεση “y = a+j” εκτελείται για δεύτερη φορά, αλλά στο νέο χώρο δεδομένων. Η μεταβλητή a αναζητείται στο ΕΔ της M, και η τιμή της που διαβάζεται από τη στοίβα είναι πάλι 1, αφού δε μεσολάβησε ανάθεση σε αυτή. Η αποτίμηση της j γίνεται τώρα με εκτέλεση του κώδικα “i+j” στο καλούν περιβάλλον, δηλαδή αυτό της προηγούμενης κλήσης της N. Στο περιβάλλον αυτό, η i είναι παράμετρος κατ’ αξία με τιμή 1, ενώ η j είναι παράμετρος κατ’ όνομα, κι επομένως αποτιμάται με εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της M, που με τη σειρά του εκτελεί τον κώδικα “a+b” στο περιβάλλον του P. Η τιμή που προκύπτει για την j είναι η 1+(-1+4+(4-2))=6. Η ανάθεση της τιμής 1+6=7 γίνεται στη μεταβλητή a του P.
Στη νέα εκτέλεση της εντολής “if”, η έκφραση “i>=0” αποτιμάται πάλι ως “Αληθής”, κι επομένως εκτελείται ξανά η πρώτη από τις δύο εντολές επιστροφής, με ΤΑ που προκύπτει από την έκφραση γινομένου με πρώτο όρο τη μεταβλητή j και δεύτερο όρο το αποτέλεσμα νέας αναδρομικής κλήσης της N. Η αποτίμηση της j γίνεται με εκτέλεση του κώδικα “i+j” στο καλούν περιβάλλον, όπου η i έχει τιμή 1, ενώ η j αποτιμάται με εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της M, που πάλι εκτελεί τον κώδικα “a+b” στο περιβάλλον του P. Η τιμή που προκύπτει για την j είναι η 1+(-1+7+(7-2))=12. Για τη νέα αναδρομική κλήση της N από την άλλη μεριά, η πρώτη παράμετρος έχει τιμή που προκύπτει από την αποτίμηση της έκφρασης “i-1”, ενώ η δεύτερη παράμετρος αποτελεί δείκτη σε κώδικα που αποτιμά την έκφραση “i+j” στο τρέχον περιβάλλον της N. Η τιμή -1 και ο δείκτης στον κώδικα “i+j” τοποθετούνται στη στοίβα. Η τιμή του ΣΠ θα είναι πάλι η διεύθυνση 972. Ένα τρίτο ΕΔ της N ενεργοποιείται, με μείωση του δείκτη στοίβας κατά 20, καλύπτοντας το χώρο διευθύνσεων [912..931]. Ο νέος συνολικός χώρος δεδομένων θα έχει τη στιγμή αυτή την ακόλουθη μορφή:

b:	-2	
a:	7	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	
i:	1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(952)
j:	“i+j”	
i:	0	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(932)
j:	“i+j”	

i:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(912)

Στην τρίτη εκτέλεση της ανάθεσης “y = a+j”, η μεταβλητή a αναζητείται ξανά στο ΕΔ της M, με τιμή που διαβάζεται από τη στοίβα πάλι ίση με 1. Η αποτίμηση της j γίνεται με εκτέλεση του κώδικα “i+j” στο καλούν περιβάλλον, στο οποίο η i έχει τιμή 0 και η j αποτιμάται με εκτέλεση του κώδικα “i+j” στο προηγούμενο περιβάλλον της N, όπου η i έχει τιμή 1, ενώ η j αποτιμάται με εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της M, που εκτελεί τον κώδικα “a+b” στο περιβάλλον του P. Η τιμή που προκύπτει για την j είναι η 0+(1+(-1+7+(7-2)))=12, ενώ η τιμή 1+12=13 ανατίθεται στη μεταβλητή a του P.
Στην εκτέλεση της εντολής “if”, η έκφραση “i>=0” αποτιμάται ως “Ψευδής”, οπότε εκτελείται η δεύτερη από τις δύο εντολές επιστροφής, τερματίζοντας την αναδρομή. Η ΤΑ προκύπτει από την αποτίμηση της μεταβλητής j, η οποία, ως παράμετρος κατ’ όνομα, αποτιμάται με εκτέλεση του κώδικα “i+j” στο καλούν περιβάλλον, κάτι που οδηγεί σε εκτέλεση του κώδικα “i+j” στο προηγούμενο περιβάλλον της N, που με τη σειρά του οδηγεί σε εκτέλεση του κώδικα “x+y+z” στο περιβάλλον της M, που πάλι εκτελεί τον κώδικα “a+b” στο περιβάλλον του P. Η τελευταία τιμή που προκύπτει για την j είναι η 0+(1+(-1+13+(13-2)))=24. Η τιμή αυτή επιστρέφεται από την τρίτη κλήση της N, κι επομένως αμέσως πριν την αύξηση του δείκτη στοίβας και την έξοδο από τη συνάρτηση N ο συνολικός χώρος δεδομένων θα έχει τη μορφή:

b:	-2	
a:	13	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	
i:	1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(952)
j:	“i+j”	
i:	0	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(932)
j:	“i+j”	
i:	-1	
ΤΑ:	24	
ΔΕ:		
ΣΠ:	972	(912)

Το πιο πρόσφατο ΕΔ της N απενεργοποιείται με την αύξηση του δείκτη στοίβας, αν και κανένας μηχανισμός δεν απαγορεύει στον κώδικα να προσπελάσει θέσεις κάτω από την κορυφή της στοίβας. Μάλιστα, αυτός είναι και ο μόνος τρόπος για ανάγνωση της ΤΑ από την προηγούμενη κλήση της N!

Μετά την έξοδο από την τρίτη κλήση της Ν, η δεύτερη κλήση της μπορεί να υπολογίσει το γινόμενο που θα δώσει τη νέα ΤΑ, που θα είναι 12*24=288. Έτσι, αμέσως πριν την επόμενη αύξηση του δείκτη στοίβας και την έξοδο από τη συνάρτηση Ν, ο συνολικός χώρος δεδομένων θα έχει τη μορφή:

b:	-2	
a:	13	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	
i:	1	
ΤΑ:		
ΔΕ:		
ΣΠ:	972	(952)
j:	“i+j”	
i:	0	
ΤΑ:	288	
ΔΕ:		
ΣΠ:	972	(932)

Το δεύτερο ΕΔ της Ν απενεργοποιείται με την αύξηση του δείκτη στοίβας στην τιμή 952 και η δεύτερη κλήση της Ν ολοκληρώνεται.
Η πρώτη κλήση της Ν μπορεί τώρα να υπολογίσει την ΤΑ, που θα είναι 5*288=1440. Αμέσως πριν την επόμενη αύξηση του δείκτη στοίβας και την οριστική έξοδο από τη συνάρτηση Ν, ο χώρος δεδομένων θα έχει τη μορφή:

b:	-2	
a:	13	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:		
ΔΕ:		
ΣΠ:	1000	(972)
j:	“x+y+z”	
i:	1	
ΤΑ:	1440	
ΔΕ:		
ΣΠ:	972	(952)

Το ΕΔ της Ν απενεργοποιείται και η Μ μπορεί να επιστρέψει την τιμή που λαμβάνει από τη συνάρτηση Ν, δηλαδή την τιμή 1440. Αμέσως πριν την αύξηση του δείκτη στοίβας και την έξοδο από τη συνάρτηση Μ, ο χώρος δεδομένων θα έχει τη μορφή:

b:	-2	
a:	13	(1000)
a:	1	
z:	“a+b”	
y:	&P::a	
x:	-1	
ΤΑ:	1440	
ΔΕ:		
ΣΠ:	1000	(972)

Με την απενεργοποίηση του ΕΔ της Μ και την επιστροφή στο πρόγραμμα Ρ μπορεί να ολοκληρωθεί η εντολή “print”. Η τιμή που εκτυπώνεται θα είναι επομένως η 1440.
Πριν την έξοδο από το πρόγραμμα Ρ, ο χώρος δεδομένων θα είναι:

b:	-2	
a:	13	(1000)