



Ανάπτυξη Μεθοδολογίας Σχεδιασμού Βέλτιστων Επεξεργαστών Ειδικού Σκοπού

ΠΕΝΕΔ-2003

Τίτλος:	Τεχνικές Ανάλυσης των Εφαρμογών με Χρήση Υποδομής Μεταγωγτιστή
Συγγραφείς:	Νικόλαος Καββαδίας και Νικόλαος Βασιλειάδης (Ερευνητές Α.Π.Θ.)
Κωδικός:	Π 2.1
Έκδοση:	1.0
Τύπος:	Παραδοτέο
Εμπιστευτικότητα:	Δημόσια
Ημερομηνία:	Νοέμβριος 18, 2006
Πρόγραμμα:	Πρόγραμμα Ενίσχυσης του Ερευνητικού Δυναμικού (ΠΕΝΕΔ 2003)
Λέξεις-Κλειδιά:	ανάλυση εφαρμογών, εφαρμογές δοκιμής, υποδομή μεταγωγτιστή, περάσματα μεταγωγτιστή, πρότυπα εργαλεία ανάπτυξης
Πρόλογος:	Αντικείμενο του παραδοτέου αυτού είναι η ανάπτυξη και τεκμηρίωση τεχνικών ανάλυσης εφαρμογών/δοκιμασίας επιδόσεων με χρήση υποδομής μεταγωγτιστή. Καταγράφεται ροή ανάλυσης η οποία χρησιμοποιείται για το χαρακτηρισμό ενσωματωμένων εφαρμογών που είναι τυπικές των προγραμματιζόμενων επεξεργαστών ειδικού σκοπού που στοχεύει η μεθοδολογία. Ο χαρακτηρισμός γίνεται με τη χρήση της υποδομής SUIF/MachSUIF στο επίπεδο της εικονικής μηχανής SUIFvm. Για το έργο αυτό, αναπτύχθηκαν περάσματα μεταγωγτιστή MachSUIF και χρησιμοποιήθηκαν διαθέσιμα υπάρχοντα. Επίσης αναπτύχθηκε κατάλληλη συντακτική διαμόρφωση για την διαχείριση της πληροφορίας των συναρτήσεων των εφαρμογών (όψεις CDFG). Τα εξαγομένα μετρικά (όπως συχνότητες εκτέλεσης, γράφοι εξάρτησης δεδομένων και εγγενής παραλληλία επιπέδου εντολών βασικών μπλοκ) πρόκειται να αξιοποιηθούν στην Φάση 3.

Ιστορικό

Ημερομηνία	Έκδοση	Σχόλια
Σεπτέμβριος 10, 2006	0.1	Πρώτη πρόχειρη έκδοση
Νοέμβριος 18, 2006	0.2	Συγκέντρωση υλικού
Δεκέμβριος 07, 2006	1.0	Επίσημη έκδοση

Πίνακας περιεχομένων

Πίνακας περιεχομένων	3
1. ΕΙΣΑΓΩΓΗ	4
1.1. Πρακτικά ζητήματα στην ανάπτυξη μιας ροής ανάλυσης εφαρμογών	4
2. ΑΡΧΙΤΕΚΤΟΝΙΚΗ SUIFVM ΚΑΙ ΦΙΛΟΙ	7
2.1. Γενικά χαρακτηριστικά της αρχιτεκτονικής SUIFvm	7
2.1.1. Το σύνολο εντολών SUIFvm	8
2.1.2. Τρόποι διευθυνσιοδότησης	10
2.1.3. Τύποι δεδομένων	11
2.1.4. Εκτέλεση/διερμηνεία εφαρμογών δοκιμής στην αρχιτεκτονική SUIFvm	11
2.1.5. Η αρχιτεκτονική SUIFvm ως «πραγματική» μηχανή	12
3. Η ΡΟΗ ΑΝΑΛΥΣΗΣ ΕΦΑΡΜΟΓΩΝ	14
3.1. Περιγραφή της ροής ανάλυσης εφαρμογών	16
3.2. Η χρήση της υποδομής μεταγλωττιστή στη ροή ανάλυσης εφαρμογών	20
3.2.1. Περάσματα ανάλυσης και βελτιστοποίησης	20
3.2.2. Βασική διαδικασία μεταγλώττισης από C σε SUIFvm	21
3.2.3. CFG τυποποίηση για τις συναρτήσεις των αναλυόμενων εφαρμογών	23
4. Η ΔΙΑΔΙΚΑΣΙΑ ΤΗΣ ΑΝΑΛΥΣΗΣ ΤΩΣ ΕΦΑΡΜΟΓΩΝ	26
4.1. Μετρικά χαρακτηρισμού εφαρμογών και διαδικασία εξαγωγής τους	27
4.2. Εφαρμογές δοκιμής	28
4.3. Πειραματικές μετρήσεις και χαρακτηρισμός των εφαρμογών	28
4.3.1. Επίδραση των βελτιστοποιήσεων μεταγλωττιστή στις επιδόσεις των εφαρμογών	28
4.3.2. Μίγμα στατικών εντολών (SIMIX) και μίγμα δυναμικών εντολών (DIMIX)	30
4.3.3. Ανάλυση τύπων δεδομένων	31
4.3.4. Εξαγωγή και χαρακτηρισμός των κρίσιμων βασικών μπλοκ των εφαρμογών	33
4.3.5. Οπτικοποίηση των γράφων εξάρτησης δεδομένων (DDG) για κρίσιμα βασικά μπλοκ των εφαρμογών	39
5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ	43
5. ΑΝΑΦΟΡΕΣ	45

1. ΕΙΣΑΓΩΓΗ

Στα πλαίσια της Φάσης 2, οι ερευνητές Α.Π.Θ. (Νικόλαος Βασιλειάδης και Νικόλαος Καββαδίας) ανέπτυξαν ροή ανάλυσης εφαρμογών (application analysis) για την εξαγωγή χρήσιμων μετρικών για ενσωματωμένες εφαρμογές που είναι τυπικές των στοχευόμενων από ειδικούς επεξεργαστές που μπορούν να σχεδιαστούν κατά τη μεθοδολογία. Η ροή ανάλυσης μπορεί εύκολα να αξιοποιηθεί και για τη δοκιμασία επιδόσεων (benchmarking) των ίδιων εφαρμογών πάνω σε αρχιτεκτονική επεξεργαστή βάσης (base processor) του οποίου επέκταση/διαφοροποίηση θα αποτελούν οι ειδικοί επεξεργαστές (ASIP/RISP) της μεθοδολογίας. Στο παραδοτέο αυτό γίνεται η καταγραφή και τεκμηρίωση της ροής ανάλυσης εφαρμογών καθώς και αποτελεσμάτων της διαδικασίας χαρακτηρισμού επιλεγμένων εφαρμογών.

1.1. Πρακτικά ζητήματα στην ανάπτυξη μιας ροής ανάλυσης εφαρμογών

Η ροή σχεδιασμού (ενότητα 3) έχει αναπτυχθεί γύρω από μια υποδομή μεταγλωττιστή (compiler infrastructure) στοχεύοντας έναν αντιπροσωπευτικό απλό επεξεργαστή, τόσο για την διαδικασία χαρακτηρισμού εφαρμογών όσο και για να αποτελέσει βάση ανάπτυξης των ειδικών επεξεργαστών. Η χρήση ενός αντιπροσωπευτικού επεξεργαστή (ή αρχιτεκτονικής-βάσης όπως λέγεται συνήθως) εξασφαλίζει κάποια ελάχιστη σταθερότητα για το περιβάλλον ανάπτυξης εφαρμογών (software development toolchain) που θα πρέπει να είναι διαθέσιμο για κάποιον τελικό ειδικό επεξεργαστή. Καθώς οι στοχευόμενες εφαρμογές έχουν αναπτυχθεί σε γλώσσα προγραμματισμού υψηλού επιπέδου (HLL) η χρήση μεταγλωττιστή είναι απαραίτητη για την εξαγωγή κώδικα συμβολομεταφραστή που θα χρησιμοποιηθεί για τον προγραμματισμό της αρχιτεκτονικής βάσης. Σχεδόν στο σύνολό τους, οι εφαρμογές των πεδίων που κυρίως ενδιαφέρουν στις Φάσεις του προγράμματος: πολυμέσα (multimedia), επεξεργασίας ψηφιακού σήματος (DSP), διαχείρισης δεδομένων δικτύου (network), καταγράφονται ως πηγαίος κώδικας στην γλώσσα προγραμματισμού ANSI C. Η χρήση μεταγλωττιστή όμως δεν είναι αρκετή καθώς (εν γένει) δεν καλύπτει τα εξής:

- δημιουργία του επιμέρους αντικείμενου κώδικα (object code) για κάθε αρχείο συμβολομεταφραστή,
- σύνδεση των επιμέρους αντικείμενων αρχείων στο τελικό εκτελέσιμο αρχείο,
- λειτουργίες εκσφαλμάτωσης (debugging) του εκτελέσιμου αρχείου για την αναζήτηση λαθών,
- επαρκής μοντελοποίηση του στοχευόμενου επεξεργαστή ώστε να είναι δυνατή η εκτέλεση του προγράμματος για το σύνολο εντολών και λαμβάνοντας υπ' όψη (μικρο)αρχιτεκτονικές λεπτομέρειες του επεξεργαστή,
- προσομοίωση του τελικού εκτελέσιμου με ακρίβεια εντολής ή/και ακρίβεια κύκλου μηχανής πάνω σε μοντέλο επεξεργαστή.

Έτσι, στη Φάση 2 επιβάλλεται η ολοκλήρωση και των συμπληρωματικών εργαλείων ανάπτυξης εφαρμογών τουλάχιστον για τον επεξεργαστή βάσης. Μία σύμβαση την οποία υιοθετούμε εδώ είναι ότι ο επεξεργαστής βάσης είναι αρχιτεκτονικής RISC, και υποστηρίζει λέξεις δεδομένων και διευθύνσεων εύρους 32-bit. Η σύμβαση αυτή είναι απαραίτητη καθώς:

- το έργο της ανάπτυξης (ή έστω και διατήρησης) ενός backend μεταγλωττιστή για έναν RISC επεξεργαστή είναι ευκολότερο από ότι για ένα CISC (σύνθετου συνόλου εντολών).
- Οι απλές εντολές του RISC μπορούν συνήθως να ταυτιστούν «ένα-προς-ένα» με τους τελεστές (π.χ. αριθμητικούς, λογικούς, διακλάδωσης/άλματος υπό συνθήκη ή

χωρίς) των συνήθων γλωσσών προγραμματισμού που βρίσκουν χρήση στα ενσωματωμένα συστήματα (ANSI C κυρίως αλλά και C++, Java, BASIC, Modula-2, Pascal).

- Η πλειοψηφία των σύγχρονων επεξεργαστών έχουν στοιχεία RISC (αρχιτεκτονική διοχέτευσης, ορθογώνια κωδικοποίηση εντολών, ομογενής αρχιτεκτονική καταχωρητών) ακόμη και αν προέρχονται από παλιές CISC οικογένειες (Itanium, Coldfire/68k).

Έχοντας υπ' όψη τα παραπάνω αλλά και την αιτιολόγηση του παραδοτέου Π1.1, αποφασίστηκε η χρήση της υποδομής SUIF/MachSUIF [SUIF],[MachSUIF] αρκετά νωρίς στα πλαίσια της έρευνας. Ο SUIF (έκδοση 2.2.0-4) χρησιμοποιείται κυρίως ως frontend στον Machine-SUIF για τον υποβιβασμό από C σε ενδιάμεση αναπαράσταση (IR) υψηλού επιπέδου, η οποία ονομάζεται High-SUIF. Η High-SUIF υποστηρίζει οντότητες όπως κόμβοι FOR, IF-THEN-ELSE και τα πρότυπα απλών εντολών τριών διευθύνσεων (Three Address Code – TAC). Ο Machine-SUIF είναι μια επεκτάσιμη υποδομή για την κατασκευή backend μεταγλωττιστών και έρχεται με ένα σύνολο περασμάτων βελτιστοποίησης και ανάλυσης [MachSUIF], υποστηρίζει δε ως είσοδο την αναπαράσταση High-SUIF την οποία υποβιβάζει σε Low-SUIF ή αλλιώς SUIFvm συμβολική assembly (εικονική μηχανή SUIF). Το SUIFvm έχει την διττή ιδιότητα να αποτελεί IR πάνω στην οποία λειτουργούν τα περάσματα βελτιστοποίησης και ανάλυσης που είναι συμβατά (δηλ. έχουν γραφεί σύμφωνα με το API) του Machine-SUIF αλλά και μια (ελλιπή όπως θα δούμε παρακάτω) αρχιτεκτονική RISC που μπορεί να χρησιμοποιηθεί ως αρχιτεκτονική βάση για την ανάλυση εφαρμογών. Ο Machine-SUIF αποδείχτηκε και ιδιαίτερα χρήσιμος για την «κατά προσέγγιση» εκτίμηση επιδόσεων των εφαρμογών όπως θα αναφερθεί παρακάτω.

Εδώ πρέπει να σημειωθεί ένα μεγάλο μειονέκτημα του περιβάλλοντος Machine-SUIF που είναι η έλλειψη αυτοματοποιήσεων για την ανάπτυξη backend για νέες αρχιτεκτονικές. Τέτοιες αυτοματοποιήσεις θα μπορούσαν να είναι οι εξής:

- Καταγραφή των κανόνων επιλογής κώδικα (code selection) με κάποιου είδους γραμματική (όπως γραμματική ιδιοτήτων)
- Χρήση παραμετροποιήσιμου κατανεμητή καταχωρητών (register allocator)
- Χρήση παραμετροποιήσιμου χρονοδρομολογητή εντολών (instruction scheduler), κάτι που φαίνεται πως υπήρχε στην πρώτη έκδοση του MachSUIF (ως 1.1.2) αλλά δεν συμπεριλαμβάνεται στον 2.02.07.15.

Χαρακτηριστικά, η ανάπτυξη του ARM backend [EPFL-ARM],[The05] για το Machine-SUIF, απαίτησε δύο εξαμηνιαίες εργασίες χωρίς με το πέρασμα της 2ης το backend αυτό να είναι ελεύθερο από σοβαρά bug (π.χ. μη υποστήριξη συναρτήσεων μεταβλητού αριθμού ορισμάτων όπως αυτές καθορίζονται από την τυποποιημένη C βιβλιοθήκη). Επίσης, ο κώδικας που μπορεί να επαναχρησιμοποιηθεί από backend σε backend (πέραν των συναρτήσεων που είναι προσβάσιμες κατά το MachSUIF API) είναι πολύ περιορισμένος. Μία επιτυχημένη υποδομή μεταγλωττιστών οφείλει να ικανοποιεί έναν εμπειρικό κανόνα: μικρές διαφορές στην αρχιτεκτονικές, μικρές και στον μεταγλωττιστή (backend). Κάτι τέτοιο ισχύει για τον LCC [LCC], εν μέρει για τον GCC [GCC] και καθόλου για τον MachSUIF.

Για την «κατά προσέγγιση» εκτίμηση επιδόσεων που αναφέρθηκε παραπάνω, ερευνητής Α.Π.Θ. (Ν. Καβαθιάς) ανέπτυξε πέρασμα για Machine-SUIF το οποίο εξάγει γράφους εξάρτησης δεδομένων (DDG) για κάθε βασικό μπλοκ μιας εφαρμογής. Σε κάθε DDG γίνεται τοπολογική ταξινόμηση για την χρονοδρομολόγηση των εντολών-κόμβων, και έτσι είναι δυνατή η εκτίμηση των κύκλων μηχανής που χρειάζονται για την σειριακή εκτέλεση τους μη

λαμβάνοντας όπως υπ' όψη φαινόμενα (π.χ. λόγω της αρχιτεκτονικής διοχέτευσης) που θα επέφεραν απώλεια σε χρήσιμους κύκλους. Η «κατά προσέγγιση» εκτίμηση επιδόσεων αποτυπώνει το άνω όριο επιδόσεων (ελάχιστο αριθμό κύκλων εκτέλεσης) της εφαρμογής σε ένα SUIFvm επεξεργαστή. Η δυνατότητα της «κατά προσέγγισης» εκτίμησης χρησιμοποιείται στην ροή ανάλυσης εφαρμογών από τους ερευνητές Α.Π.Θ.

Ακριβέστερη εκτίμηση επιδόσεων είναι εφικτή μόνο με τη χρήση προσομοιωτών ακρίβειας κύκλου. Στην πραγματικότητα, η ανάπτυξη εφαρμογών γίνεται με χρήση «ζεύγους» προσομοιωτών: ενός ακρίβειας εντολής και ενός ακρίβειας κύκλου. Ο προσομοιωτής ακρίβειας εντολής χρησιμοποιείται για να διαπιστωθεί η ορθή εκτέλεση του εκτελέσιμου της εφαρμογής από το σύνολο εντολών που υποστηρίζει ο επεξεργαστής. Ο προσομοιωτής ακρίβειας κύκλου μοντελοποιεί την μικροαρχιτεκτονική του επεξεργαστή (λεπτομέρειες όπως τα στάδια διοχέτευσης, εντολές πολλαπλών κύκλων κ.λ.π.) και αποτυπώνει την συμπεριφορά του σε κάθε κύκλο μηχανής. Προκειμένου την ανάπτυξη των απαιτούμενων προσομοιωτών ακρίβειας κύκλου επιλέχθηκε το περιβάλλον ανάπτυξης προσομοιωτών ArchC [ArchC],[Aze05]. Κεντρικό ρόλο στο περιβάλλον ArchC έχει η γλώσσα περιγραφής αρχιτεκτονικής ArchC η οποία προσφέρει ένα υψηλότερο επίπεδο αφαίρεσης από την SystemC [SystemC] και διευκολύνει την περιγραφή αρχιτεκτονικών επεξεργαστών και αυτόματη παραγωγή του αντίστοιχου SystemC κώδικα. Η διαθεσιμότητα μεγάλου αριθμού μοντέλων από τους συντάκτες του περιβάλλοντος ArchC για γνωστούς επεξεργαστές (MIPS, SPARC, PowerPC, 8051, PIC κ.α.) βοήθησε σημαντικά στην εξοικείωση των ερευνητών.

Αξίζει να σημειωθεί εδώ ότι στα πλαίσια της αναζήτησης περιβάλλοντος ανάπτυξης προσομοιωτών εξετάστηκαν και άλλες λύσεις, ιδιαίτερα τα περιβάλλοντα SimpleScalar [SimpleScalar],[Bur97],[Aus02] και Processor Designer [Cware], [Hof02]. Ο SimpleScalar αποτελεί μια δημοφιλή σουίτα παραμετρικών μικροαρχιτεκτονικών προσομοιωτών ανοικτού κώδικα για μια υπερβαθμωτή παραλλαγή μιας αρχιτεκτονικής MIPS R3000. Για το είδος της έρευνας μας ο SimpleScalar δεν είναι η καλύτερη επιλογή για τους εξής λόγους:

- μη πρακτικός φορμαλισμός για την καταγραφή του συνόλου εντολών. Χρησιμοποιούνται μακροεντολές στην C, που ανήκουν σε ένα είδος «γλώσσας» προεπεξεργασίας με όλα τα σχετικά μειονεκτήματα (το κυριότερο από αυτά να είναι η έλλειψη τυποποίησης και αντίστοιχων εργαλείων συντακτικής ανάλυσης)
- δεν επιτρέπει εύκολα μεταβολές του αρχιτεκτονικού μοντέλου (π.χ. αριθμός σταδίων διοχέτευσης) και δεν επιτρέπει τη δημιουργία ενός αυθαίρετου μοντέλου
- κάνει εκτίμηση αριθμού κύκλων και δεν καταγράφει την ακριβή κατάσταση (state) των σταδίων διοχέτευσης του επεξεργαστή σε κάθε κύκλο
- έχει κωδικοποίηση εντολής μήκους 64-bit κάτι που αυξάνει τις απαιτήσεις σε επιφάνεια για ένα επεξεργαστή που θα σχεδιαζόταν στη βάση του SimpleScalar
- δεν επιτρέπει την προσομοίωση πλατφόρμας επεξεργαστή, δηλαδή μοντέλο επεξεργαστή μαζί με περιφερειακές μονάδες συνδεδεμένα με αρχιτεκτονική διαύλου (κάτι που αρχίζει να γίνεται πράξη στην ArchC 2.0)

Ο Cware Processor Designer [Cware],[Hof02] αποτελεί ένα εμπορικό/κλειστού κώδικα περιβάλλον σχεδιασμού πλατφόρμας επεξεργαστή-εργαλείων ανάπτυξης εφαρμογών που βασίζεται στην γλώσσα περιγραφής αρχιτεκτονικής LISA 2.0. Οι ερευνητές Α.Π.Θ. έχουν στη διάθεσή τους την ακαδημαϊκή έκδοση του Processor Designer (η οποία ελλείπεται μεταγλωττιστή) ενώ τα υπόλοιπα επαναστοχευόμενα εργαλεία δεν ακολουθούν τις GNU συμβάσεις με αποτέλεσμα να είναι δύσκολη η χρήση άλλου μεταγλωττιστή για τον σχεδιαζόμενο επεξεργαστή με αυτά.

2. ΑΡΧΙΤΕΚΤΟΝΙΚΗ SUIFVM ΚΑΙ ΦΙΛΟΙ

2.1. Γενικά χαρακτηριστικά της αρχιτεκτονικής SUIFvm

Κεντρική θέση στη ροή ανάλυσης εφαρμογών που αναπτύχθηκε από τους ερευνητές Α.Π.Θ. έχει η αρχιτεκτονική SUIFvm. Η αρχιτεκτονική SUIFvm διαδραματίζει διττό ρόλο: αποτελεί την ενδιάμεση αναπαράσταση στην οποία η υποδομή SUIF/MachSUIF αποδομεί μια εφαρμογή (σε C, C++, Fortran, Java) που δέχεται ως είσοδο αλλά και μια αρχιτεκτονική ενός αφαιρετικού επεξεργαστή σε επίπεδο εικονικής μηχανής (virtual machine).

Το SUIFvm IR είναι οργανώσιμο σε διαφορετικές μορφές γράφων (Πίνακας 2.1) οι οποίες αντιμετωπίζουν τα βασικά μπλοκ ως κόμβους (nodes): τέτοιοι είναι οι γράφοι ροής ελέγχου (CFG) και τα CFG με στατικές απλές αναθέσεις (SSA) [Cyt91]. Οι δύο αυτοί τύποι υποστηρίζονται από την επίσημη διανομή του MachSUIF. Σε κάθε συνάρτηση μιας εξεταζόμενης εφαρμογής αντιστοιχεί ένας γράφος CFG ή CFG-SSA. Τρίτοι ερευνητές έχουν προτείνει (και υλοποιήσει) την οργάνωση του SUIFvm IR σε μορφή στατικής απλής πληροφορίας (SSI) [Sin04] και CCDG (Συντεθειμένος Γράφος Εξάρτησης Ελέγχου) [She04], με το τελευταίο να είναι ένας ενδιαφέρων τύπος Γράφου Εξάρτησης Προγράμματος (PDG) [Fer87]. Αναπαραστάσεις τύπου PDG έχουν το πλεονέκτημα της καταγραφής της συμπεριφοράς μιας εφαρμογής από έναν μόνο κατάλληλα ιεραρχημένο γράφο, κάτι που επιτρέπει την εφαρμογή βελτιστοποιήσεων σε ολόκληρο το πρόγραμμα (τεχνικές WPO). Καθώς όμως τα περισσότερα περάσματα βελτιστοποίησης και ανάλυσης τόσο της διανομής του MachSUIF όσο και των ερευνητών Α.Π.Θ. δέχονται τους CFG και SSA τύπους, δεν εξετάστηκε περαιτέρω η αξιοποίηση του υλικού που αναφέρεται για τις αναπαραστάσεις τύπου SSI και CCDG.

Πίνακας 2.1: Διαφορετικοί τύποι γράφων ροής ελέγχου στους οποίους είναι οργανώσιμη η ενδιάμεση αναπαράσταση SUIFvm.

Είδος γράφου	Περιγραφή της Αναπαράστασης		Αναφορά
CFG	Γράφος Ροής Ελέγχου		[MachSUIF]
CFG-SSA	CFG Στατικής Απλής Ανάθεσης		[MachSUIF]
SSI	Στατική Απλή Πληροφορία		[Sin04]
CCDG	Συντεθειμένος Γράφος Εξάρτησης Ελέγχου		[She04]

Ιδιαίτερη σημασία στην ανάπτυξη της ροής ανάλυσης εφαρμογών της Φάσης 2 έχει το γεγονός ότι το SUIFvm ερμηνεύεται ως σύνολο (σετ) εντολών μιας (ατελούς) αρχιτεκτονικής επεξεργαστή. Η αρχιτεκτονική SUIFvm διαθέτει ένα σύνολο εντολών με χαρακτηριστικά κλασσικού RISC (κοντά στο πνεύμα των αρχιτεκτονικών MIPS [MIPS], DLX [Hen96], PowerPC, Nios 2 [Nios-II]) με τη βασική διαφορά ότι υποστηρίζει απεριόριστους εικονικούς καταχωρητές (virtual registers). Στο επίπεδο SUIFvm μπορούν να εφαρμοστούν περάσματα βελτιστοποίησης κώδικα (για το MachSUIF ή το SUIF για την βελτιστοποίηση του High-SUIF) που έχουν αντίκτυπο στην ποιότητα κώδικα SUIFvm. Τέτοια περάσματα του MachSUIF είναι τα εξής: για την εξουδετέρωση νεκρού κώδικα, εξουδετέρωση κοινής υποεκφράσεως, μετατροπή δήλωσης «αν» (if-conversion), και κίνηση κώδικα (code motion) [EPFL-passes].

2.1.1. Το σύνολο εντολών SUIFvm

Το σύνολο εντολών της αρχιτεκτονικής εικονικής μηχανής SUIFvm δίνεται αναλυτικά στον Πίνακα 2.2. Το σετ εντολών SUIFvm έχει στοιχεία αρχιτεκτονικής φόρτωσης-αποθήκευσης (εντολές LOD, STR). Η εντολή MEMCPY που μετακινεί δεδομένα ανάμεσα σε μνήμες είναι μεν CISC χαρακτηριστικό, όμως δεν υποστηρίζεται η αυτόματη επιλογή της κατά την μετατροπή από High-SUIF σε Low-SUIF (και SUIFvm), η οποία είναι μια διαδικασία απλοϊκής επιλογής κώδικα. Επιπλέον παρατηρήσεις πάνω στο SUIFvm σετ εντολών έχουν ως εξής:

- οι δυαδικές (ήτοι δεχόμενες δύο ορίσματα πηγής) αριθμητικές (Α) και λογικές εντολές (Λ), όπως και οι εντολές ολίσθησης (Ο) και σύγκρισης (Σ) δέχονται ως ορίσματα είτε δύο καταχωρητές, είτε ένα καταχωρητή και ένα άμεσο όρισμα (immediate). Επειδή ο μεταγλωττιστής (ατυχώς) επιτρέπει και το σχηματισμό δυαδικών εντολών με δύο άμεσα ορίσματα, τέτοιες περιπτώσεις ανιχνεύονται από πέρασμα διπλώσης και διάδοσης σταθεράς (constant folding/propagation) και εξουδετερώνονται.
- Η MOD δεν χρησιμοποιείται.
- Η MBR (διακλάδωση πολλαπλών επιλογών) προκύπτει από τον υποβιβασμό μιας δήλωσης switch από την ANSI C. Λόγω του ότι διαθέτει μεταβλητό αριθμό ορισμάτων, και η κωδικοποίηση μιας MBR n διευθύνσεων-στόχων θα απαιτούσε n λέξεις μηχανής, η χρήση της αποφεύγεται. Η αποδόμησή της σε απλές εντολές διακλάδωσης γίνεται από την αλληλουχία περασμάτων που κωδικοποιεί το script **do_lower**.
- Η εντολή κλήσης υπορουτίνας (CAL) επίσης δέχεται μεταβλητό αριθμό ορισμάτων.
- Οι εντολές ABS, MIN, MAX είναι σχετικά υψηλού επιπέδου και όπως η MEMCPY δεν είναι αυτόματα αναγνωρίσιμες (επιλέξιμες) από το frontend.
- Οι εντολές ANY και MRK χρησιμοποιούνται εσωτερικά από το μεταγλωττιστή και δεν αντιστοιχούν σε πραγματικές εντολές στο επίπεδο μηχανής.

Πίνακας 2.2: Το σετ εντολών SUIFvm.

Μνημονικό	Τύπος εντολής	Περιγραφή	Σύνταξη (BNF)
NOP	--	Καμία λειτουργία	nop
LDA	A	Φόρτωσης άμεσης διεύθυνσης	lda rdst, symbol
LDC	A	Φόρτωση σταθεράς	ldc rdst, immed
LOD	M	Φόρτωση καταχωρητή από διεύθυνση στη μνήμη	lod rdst, addr
STR	M	Αποθήκευση καταχωρητή σε διεύθυνση στη μνήμη	str rsrc, addr
MEMCPY	M	Αντιγραφή τιμής από διεύθυνση στη μνήμη σε διεύθυνση στη μνήμη	memcpy addrsrc, addrdst
MOV	A	Μεταφορά από καταχωρητή σε καταχωρητή	mov rdst, rsrc
CVT	--	Μετατροπή τύπου δεδομένων και μεταφορά από καταχωρητή σε καταχωρητή	cvt rdst, rsrc
NEG	A	Συμπλήρωμα ως προς 2	neg rdst, rsrc
ADD	A	Άθροιση	add rdst, rsrc1, op2
SUB	A	Αφαίρεση	sub rdst, rsrc1, op2



MUL	A(Π)	Πολλαπλασιασμός	add rdst, rsrc1, op2
DIV	A(Δ)	Διαίρεση	add rdst, rsrc1, op2
REM	A(Δ)	Ακέραιο υπόλοιπο (όπως ο τελεστής % στην ANSI C)	add rdst, rsrc1, op2
MOD	A(Δ)	Ακέραιο υπόλοιπο (επιστρέφει πάντα θετική τιμή)	add rdst, rsrc1, op2
NOT	Λ	Λογικό ΟΧΙ	not rdst, rsrc
AND	Λ	Λογικό ΚΑΙ	and rdst, rsrc1, op2
IOR	Λ	Λογική Ή	ior rdst, rsrc1, op2
XOR	Λ	Λογικό αποκλειστικό-Ή	xor rdst, rsrc1, op2
ASR	O	Αριθμητική ολίσθηση προς τα δεξιά	asr rdst, rsrc1, op2
LSR	O	Λογική ολίσθηση προς τα δεξιά	lsr rdst, rsrc1, op2
LSL	O	Λογική ολίσθηση προς τα αριστερά	lsl rdst, rsrc1, op2
MIN	A	Εύρεση ελάχιστης τιμής	min rdst, rsrc1, op2
MAX	A	Εύρεση μέγιστης τιμής	max rdst, rsrc1, op2
ABS	A	Απόλυτη τιμή	abs rdst, rsrc
ROT	O	Κυκλική ολίσθηση	rot rdst, rsrc1, op2
SEQ	Σ	Σύγκριση ισότητας	seq rdst, rsrc1, op2
SNE	Σ	Σύγκριση ανισότητας	sne rdst, rsrc1, op2
SL	Σ	Σύγκριση μικρότερο από	sl rdst, rsrc1, op2
SLE	Σ	Σύγκριση μικρότερο ή ίσο από	sle rdst, rsrc1, op2
JMP	Δ1	Άλμα χωρίς συνθήκη	jmp target
JMPI	Δ1	Έμμεσο άλμα χωρίς συνθήκη	jmp rsrc
BTRUE	Δ2	Διακλάδωση αν η τιμή καταχωρητή είναι διάφορη του μηδενός	btrue rsrc, target
BFALSE	Δ2	Διακλάδωση αν η τιμή καταχωρητή είναι ίση με το μηδέν	bfalse rsrc, target
BEQ	Δ2	Διακλάδωση αν ίσο	beq rsrc1, rsrc2, target
BNE	Δ2	Διακλάδωση αν διαφορετικό από	bne rsrc1, rsrc2, target
BGE	Δ2	Διακλάδωση αν μεγαλύτερο ή ίσο από	bge rsrc1, rsrc2, target
BGT	Δ2	Διακλάδωση αν μεγαλύτερο από	bgt rsrc1, rsrc2, target
BLE	Δ2	Διακλάδωση αν μικρότερο ή ίσο από	ble rsrc1, rsrc2, target
BLT	Δ2	Διακλάδωση αν μικρότερο από	blt rsrc1, rsrc2, target
MBR	Δ2	Διακλάδωση με πολλαπλές διευθύνσεις στόχους	mbr rsrc, target1{, targetn}+ με n>=2
CAL	Υ	Κλήση υπορουτίνας	cal rdst{, rsrcn}+ με n>=2
RET	Υ	Επιστροφή από υπορουτίνα	ret {rsrc}?
ANY	--	Εντολή χωρίς προκαθορισμένη λειτουργία	any --
MRK	--	Εντολή για την εισαγωγή σημειώσεων στον κώδικα συμβολομεταφραστή	mrk --

Υπόμνημα 2.2.1:

Συντόμηση	Σημασία = Τύπος εντολής	Εναλλακτικός συμβολισμός
A, (Π, Δ)	Αριθμητική. (Π) για πολλαπλασιασμού και	arith (multiply, division)

	(Δ) για διαίρεσης (υποκατηγορίες)	
Λ	Λογική	logical
Σ	Σύγκρισης	set
Ο	Ολίσθησης	shift
Μ	Μεταφοράς δεδομένων από/προς τη μνήμη	mem
Δ1	Διακλάδωσης/Άλματος χωρίς συνθήκη	ubr
Δ2	Διακλάδωσης/Άλματος υπό συνθήκη	cbr
Υ	Διαχείρισης υπορουτίνας	call/ret
--	Άλλο	other

Υπόμνημα 2.2.2:

Συντόμηση	Σημασία
r[dst src[n]]	Καταχωρητής προορισμού ή πηγής με α.α. <i>n</i>
op2	Όρισμα πηγής με α.α. 2. που δίνεται από τη σύνταξη: rsrc2 immed
immed	Άμεσο όρισμα
symbol	Σύμβολο
addr[src dst]?	Έκφραση διεύθυνσης
target{n}	Διεύθυνση στόχος
--	Δεν έχει προκαθοριστεί

2.1.2. Τρόποι διευθυνσιοδότησης

Η αρχιτεκτονική SUIF_{vm} όπως καθορίζεται στην τελευταία δημόσια έκδοση του MachSUIF (2.02.07.15) υποστηρίζει τους επτά διαφορετικούς τρόπους διευθυνσιοδότησης (addressing modes) του Πίνακα 2.3, οι οποίοι μπορούν να χρησιμοποιηθούν για την υλοποίηση σύνθετων τρόπων διευθυνσιοδότησης σε backend για κατάλληλους στοχευόμενους επεξεργαστές. Με ADDR_SYM συμβολίζεται η άμεση διευθυνσιοδότηση από συμβολική σταθερά, η οποία αν και είναι όρισμα (τύπος Opnd) κατά SUIF_{vm} και όχι τρόπος διευθυνσιοδότησης, χρησιμοποιείται για τη προσπέλαση μεταβλητών συνολικής και τοπικής εμβέλειας. Στην πράξη, η διαδικασία επιλογής SUIF_{vm} κώδικα χρησιμοποιεί μόνο τους τρόπους ADDR_SYM και BASE_DISP.

Πίνακας 2.3: Τρόποι διευθυνσιοδότησης στο σετ εντολών SUIF_{vm}.

Αρίθμηση	Σύμβολο	Είδος	Λειτουργία
1	ADDR_SYM	Όρισμα	addr_sym
2	SYM_DISP	Τρόπος διευθυνσιοδότησης	addr_sym + disp
3	INDEX_SYM_DISP	-«-	index + addr_sym + disp
4	BASE_DISP	-«-	base + disp
5	BASE_INDEX	-«-	base + index
6	BASE_INDEX_DISP	-«-	base + index + disp
7	INDEX_SCALE_DISP	-«-	index*scale + disp
8	BASE_INDEX_SCALE_DISP	-«-	base + index*scale + disp

Υπόμνημα 2.3.1:

Συντόμηση	Σημασία
addr_sym	Συμβολική διεύθυνση
disp	Άμεσο ακέραιο όρισμα που δηλώνει μετατόπιση
index	Καταχωρητής δείκτη
base	Καταχωρητής βάσης
scale	Σταθερά κλιμάκωσης

Οι τρόποι διευθυνσιοδότησης του Πίνακα 2.3 θα μπορούσαν να αντικατασταθούν από δύο μόνο βασικούς «υπερτύπους»:

- Ο τύπος 3 (INDEX_SYM_DISP) είναι η γενική περίπτωση (υπερσύνολο) των τύπων 1-3
- Ο τύπος 8 (BASE_INDEX_SCALE_DISP) είναι υπερσύνολο των τύπων 4-8

Οι δύο προϋποθέσεις ώστε να είναι αυτό εφικτό είναι η αποδοχή σταθερών καταχωρητών με τιμή μηδέν καθώς και του «μηδενικού» (null) συμβόλου. Σταθεροί καταχωρητές (συνήθως με τιμή 0 ή 1) υποστηρίζονται από τις περισσότερες πραγματικές αρχιτεκτονικές επεξεργαστών. Δεν έχει διευκρινιστεί γιατί η ομάδα ανάπτυξης του Machine-SUIF δεν ακολούθησε αυτή την τακτική.

2.1.3. Τύποι δεδομένων

Η αρχιτεκτονική SUIFvm υποστηρίζει τους βασικούς τύπους δεδομένων του Πίνακα 2.4 για αναπαραστάσεις αριθμών ακέραιων και κινητής υποδιαστολής καθώς και για μεταβλητές δεικτοδότησης (pointers), οι οποίες εκφράζουν διευθυνσιοδότηση σε εντολές LOD και STR. Οι τύποι δεδομένων είναι ρητώς εκπεφρασμένοι στον κώδικα SUIFvm ώστε να συγκεκριμενοποιείται η σημασιολογία κάποιων εντολών όπως αυτό είναι απαραίτητο. Αυτό συμβαίνει για τις εντολές LOD, STR ώστε να γνωστοποιείται το εύρος της μεταφοράς από και προς την μνήμη δεδομένων αλλά και για την εντολή μετατροπής τύπου (CVT) για τον καθορισμό των τύπων των καταχωρητών πηγής και προορισμού. Ένας συμβολικός καταχωρητής δίνεται από την έκφραση:

$$\$vr < n > .[u | s | f | p | v] < m >$$

όπου:

- Τα “u”, “s” δίνουν το πρόσημο
- Το “f” συμβολίζει αναπαράσταση κινητής υποδιαστολής
- Το “p” την αναπαράσταση δεικτοδότησης
- Το “v” συμβολίζει τον κενό τύπο (κατά το “void” της γλώσσας C)
- $< n >$ είναι ο αύξων αριθμός του καταχωρητή
- $< m >$ είναι το εύρος bit του καταχωρητή

Πίνακας 2.4: Τύποι δεδομένων στο σετ εντολών SUIFvm.

Τύπος	Συμβολισμός	Επιτρεπόμενο εύρος bit
void (κενός)	v	0
integer (ακέραιος)	u (απρόσημοι ακέραιοι) s (προσημασμένοι ακέραιοι)	8, 16, 32, 64
floating-point (κινητής υποδιαστολής)	f	32, 64, 128
pointer (δεικτοδότησης)	p	32, 64

2.1.4. Εκτέλεση/διερμηνεία εφαρμογών δοκιμής στην αρχιτεκτονική SUIFvm

Όπως έχει ήδη αναφερθεί, η αρχιτεκτονική SUIFvm αποτελεί μια «εικονική μηχανή» υιοθετώντας συμβάσεις (απεριόριστοι καταχωρητές, πολυσημία εντολών που σημαίνει αναντιστοιχία με τις πραγματικές απαιτήσεις σε διαθέσιμα orcode) με στόχο τη διευκόλυνση της επιλογής κώδικα και εφαρμογής περασμάτων μετασχηματισμών, τα οποία είναι επιθυμητό να επαναχρησιμοποιούνται σε διαφορετικές αρχιτεκτονικές. Στο πλαίσιο της έρευνας είναι πολύ χρήσιμο να έχουμε ποσοτική αντίληψη των χαρακτηριστικών των

εφαρμογών δοκιμής στο επίπεδο SUIFvm, κάτι που απαιτεί οι εφαρμογές αυτές να μπορούν να εκτελεστούν ή να διερμηνευτούν σε προσομοιωτή του συνόλου εντολών SUIFvm. Η διαδικασία της εκτέλεσης συνήθως ακολουθείται για πραγματικές (πλήρεις) αρχιτεκτονικές επεξεργαστών καθώς απαιτεί τη δυνατότητα παραγωγής του αντίστοιχου εκτελέσιμου αρχείου. Το τελευταίο υπονοεί την αναγκαιότητα επαναστόχευσης του συμβολομεταφραστή και του συνδέτη (**as** και **ld** αντίστοιχα στο εργαλείο **binutils** [Binutils]) για τη συγκεκριμένη αρχιτεκτονική. Αυτό προϋποθέτει τον καθορισμό της κωδικοποίησης εντολών μέσα στα όρια του εύρους bit των εντολών, κάτι που δεν μπορεί να συμβεί για την SUIFvm, για την οποία οι δείκτες καταχωρητών μπορούν να είναι αυθαίρετα μεγάλοι ακέραιοι. Η δεύτερη λύση (διερμηνεία προγράμματος επιπέδου συμβολομεταφραστή) είναι η μόνη εφαρμόσιμη. Ο διερμηνευτής/προσομοιωτής θα πρέπει να δέχεται ως είσοδο αναπαράσταση SUIFvm και να την προσομοιώνει θεωρώντας έναν πρότυπο επεξεργαστή SUIFvm. Οι ερευνητές Α.Π.Θ. επέλεξαν μια λύση αυτής της κατηγορίας: συγκεκριμένα τη χρήση του C backend (πέρασμα **m2c**) το οποίο αντιστοιχίζει 1-προς-1 εντολές SUIFvm με αναθέσεις τριών διευθύνσεων (TAC) που προσδιορίζονται από ένα περιορισμένο υποσύνολο της C. Έτσι, για κάθε εφαρμογή δοκιμής, μπορεί να παραχθεί η αναπαράστασή στο υποσύνολο της C από το SUIFvm. Έπειτα, το C αυτό αρχείο μπορεί να μεταγλωττιστεί από τα εργαλεία (gcc, binutils) ενός τυπικού Linux/x86 συστήματος ανάπτυξης λογισμικού. Ο Πίνακας 2.5 δίνει τις αντιστοιχίες εντολών SUIFvm με αναθέσεις TAC σε C.

Πίνακας 2.5: Αντιστοιχία εντολών SUIFvm με αναθέσεις τριών διευθύνσεων (TAC) σε υποσύνολο της γλώσσας C.

Εντολή	Μορφή κώδικα τριών διευθύνσεων
ADD, SUB, MUL, DIV, REM, LSL, LSR, ASR, AND, IOR, XOR, MIN, MAX	<code>rdst = rsrc1 binop op2;</code>
NEG, NOT, ABS	<code>rdst = unop rsrc1;</code>
SEQ, SNE, SL, SLE	<code>rdst = rsrc1 condop2 op2;</code>
RET	<code>return rsrc;</code>
CVT	<code>rdst = (castop)rsrc;</code>
LOD	<code>rdst = *addr;</code>
STR	<code>*addr = rsrc;</code>
BFALSE, BTRUE, BEQ, BNE, BLE, BLT, BGE, BGT	<code>if (src1 condop op2) goto target;</code>
JMP	<code>goto target;</code>
CAL	<code>(rdst =)? func-name (arg1 {, argn}+);</code>

Υπόμνημα 2.5.1:

Συντόμευση	Ερμηνεία	Αντιστοίχιση
binop	Δυαδικός τελεστής	<code>+, *, /, %, >>, <<, &, , ^, min(), max()</code>
unop	Μοναδικός τελεστής	<code>!, ~, abs()</code>
condop	Τελεστής σύγκρισης	<code>==, !=, >=, >, <=, <</code>
castop	Τελεστής μετατροπής τύπου	<code>(<ANSI C τύπος>)</code>
condop2	Τελεστής σύγκρισης (υποσύνολο που χρησιμοποιείται από τις εντολές σύγκρισης)	<code>==, !=, <=, <</code>
selop	Τελεστής επιλογής (σε τρίγραφο μορφή)	<code>(συνθήκη ? τότε : αλλιώς)</code>

2.1.5. Η αρχιτεκτονική SUIFvm ως «πραγματική» μηχανή

Παρά το γεγονός ότι μπορεί να γίνει κάποια εκτίμηση επιδόσεων στο επίπεδο SUIFvm, η ανάλυση των χαρακτηριστικών εφαρμογών που εκτελούνται σε αρχιτεκτονικές επεξεργαστών με πραγματικούς περιορισμούς αποκαλύπτουν ακόμη περισσότερα στοιχεία για τη

συμπεριφορά τους αλλά και για τα επιθυμητά χαρακτηριστικά των αντίστοιχων ειδικών επεξεργαστών στους οποίους θα στοχευτούν (Φάσεις 3-5). Τέτοιοι περιορισμοί είναι για παράδειγμα οι εξής:

- Αρχιτεκτονική μνήμης: ιεραρχία κρυφών μνημών, χρονισμός των προσπελάσεων σε στοιχεία αποθήκευσης (όπως εξωτερική μνήμη)
- Πεπερασμένος αριθμός καταχωρητών
- Ετερογενής αρχιτεκτονική καταχωρητών
- Διαμορφώσεις εντολής με συγκεκριμένα εύρη bit
- Οργάνωση σταδίων διοχέτευσης
- Διακοπές (interrupts) και εξαιρέσεις (exceptions)
- Μονάδες υλικού για υποστήριξη ΛΣ (λειτουργικού συστήματος) όπως μονάδα διαχείρισης μνήμης (MMU)

χωρίς αυτή η λίστα να είναι πλήρης.

Αλλαγές προς αυτή τη κατεύθυνση έχουν υλοποιηθεί από τους ερευνητές Α.Π.Θ.. Ερευνητής Α.Π.Θ. (Ν. Καββαδίας) έχει υλοποιήσει προσομοιωτή ακρίβειας εντολής για παράγωγη αρχιτεκτονική του SUIFvm ονόματι SUIFrm (πραγματική μηχανή SUIF). Οι αλλαγές που έχουν εφαρμοστεί συνοψίζονται παρακάτω:

- Εντολή LDCH (load constant high halfword) για τη φόρτωση των άνω 16-bit μιας 32-bit σταθεράς σε καταχωρητή. Επιτρέπει τη φόρτωση σταθερών 32-bit με δύο εντολές (ldch, ior με άμεσο όρισμα). Η LDCH ακολουθεί τη σύνταξη της LDC.
- Οργάνωση του αρχείου καταχωρητών κατά το πρότυπο των DLX και MIPS R3000 οι οποίοι υποστηρίζουν ομογενές αρχείο καταχωρητών με 32 θέσεις με δείκτη στοίβας (r29), δείκτη πλαισίου (r30) και διεύθυνση επιστροφής (r31).
- Η πολυσημία της εντολής CVT η οποία περιλαμβάνει 28 διαφορετικές λειτουργίες μετατροπής (όσες είναι οι διατάξεις των 8-ανά-2) μπορεί να περιοριστεί σε τρεις τύπους: τις εντολές ZXT (επέκταση μηδενός), SXT (επέκταση πρόσημου) και TRUNC (αποκοπή από μεγάλο σε μικρό εύρος bit).

Επίσης στο σύνολο εντολών του SUIFvm, οι ερευνητές Α.Π.Θ. προσέθεσαν και την εντολή SELECT (εντολή επιλογής). Η εντολή SELECT υλοποιεί τη λειτουργία της επιλογής της ανάθεσης σε καταχωρητή προορισμού από δύο πηγαία ορίσματα με την ίδια έννοια που αυτό γίνεται από ένα πολυπλέκτη (multiplexer). Η λειτουργία της SELECT μπορεί να αποδοθεί ως εξής:

$$dst \leftarrow \text{SELECT } cond, src1, src2 \text{ όπου } dst = \begin{cases} src1 \text{ αν } cond \text{ είναι αληθές} \\ src2 \text{ αν } cond \text{ είναι ψευδές} \end{cases}$$

Οι εντολές SELECT εισάγονται στο επίπεδο SUIFvm εφόσον εφαρμοστεί πέρασμα μετασχηματισμού τύπου if-conversion [Rol03]. Ο μετασχηματισμός αυτός αντικαθιστά ορισμένες περιπτώσεις if-then-else δηλώσεων στην ANSI C από εντολές SELECT με τα κατάλληλα ορίσματα. Δηλαδή, η εφαρμογή του if-conversion έχει ως αποτέλεσμα την συνένωση βασικών μπλοκ καθώς «διαγράφονται» οι εντολές διακλάδωσης που επάγονται από την μεταγλώττιση των if-then-else σε επίπεδο SUIFvm. Είναι γνωστό τόσο από την εμπειρία των ερευνητών Α.Π.Θ. όσο και από τη βιβλιογραφία ότι μεγαλύτερα βασικά μπλοκ συνήθως προσφέρουν και περισσότερες ευκαιρίες για ανεύρεση ειδικών εντολών (Φάση 4) που μπορούν να επιταχύνουν την εκτέλεση εφαρμογών στους επεξεργαστές ειδικού σκοπού. Η προσθήκη εντολών SUIFvm επίσης ευνοεί την ανάδειξη της εγγενούς παραλληλίας

πολλών ενσωματωμένων εφαρμογών και για το λόγο αυτό μετασχηματισμοί if-conversion χρησιμοποιούνται ως επί το πλείστον από μεταγλωττιστές που στοχεύουν VLIW επεξεργαστές (π.χ. Trimaran/IMPACT) [Trimaran],[OpenIMPACT].

Μια σημαντική διαπίστωση που προέκυψε από τη διαδικασία αυτή είναι ότι μια αρχιτεκτονική SUIFrm είναι συμβατή με την απλή αρχιτεκτονική DLX στο επίπεδο συμβολομεταφραστή. Αυτό συμπεραίνεται από το γεγονός ότι υπάρχει πεπερασμένο σύνολο μετασχηματισμών που μετατρέπει κώδικα SUIFrm σε DLX και το αντίστροφο. Ο Πίνακας 2.6 συνοψίζει προτεινόμενους μετασχηματισμούς οι οποίοι μπορούν να εφαρμοστούν με τη χρήση κλασσικών εργαλείων βελτιστοποίησης «κλειδαρότρυπας» (peephole optimization) που χρησιμοποιούν την απλή τεχνική της «απομόνωσης και αντικατάστασης γραμμής» (line buffering and substitution). Ένα εύχρηστο εργαλείο αυτού του είδους είναι και ο “cort” [cort].

Πίνακας 2.6: Μετασχηματισμοί από γλώσσα συμβολομεταφραστή SUIFrm σε DLX και το αντίστροφο.

Μοτίβο (pattern) εντολών ή μνημονικό σε SUIFrm	Αντίστοιχο μοτίβο εντολών ή μνημονικό για τον DLX
LODW, LODH, LODHU, LODB, LODBU, STRW, STRH, STRB	LW, LH, LHU, LB, LBU, SW, SH, SB
JMP, JMPL, CAL(R)	J, JR, JAL(R)
BFALSE, BTRUE	BEQZ, BNEZ
LSL, LSR, ASR	SLL, SRL, SRA
IOR, MUL	OR, MULT
NEG%1 %0, %2	SUB%1 %0, \$zero, %2
LDCH %0, %1	LHI %0, %hi(%1)
IOR %0, %0, %1	ORI %0, %0, %lo(%1)
SL%3 %0, %2, %1	SGT%3 %0, %1, %2
SLE%3 %0, %2, %1	SGE%3 %0, %1, %2
opcode %0, %1, #%2	LHI \$at, %hi(%2) ORI \$at, \$at, %lo(%2) opcode %0, \$1, \$at

Υπόμνημα 6.1:

Συντόμευση	Σημασία
opcode	Εντολή με δύο πηγαία ορίσματα
\$at	«Πρόχειρος» καταχωρητής που χρησιμοποιείται από τον συμβολομεταφραστή

Πρέπει να σημειωθεί ότι υφίστανται ορισμένες περιπτώσεις όπου δεν είναι εφαρμόσιμοι απλοϊκοί μετασχηματισμοί κειμένου:

- Κώδικας DLX με RFE (επιστροφή από εξαίρεση) και TRAP (παγίδα και άλμα σε υπορουτίνα εξυπηρέτησης από διακοπή λογισμικού).
- Κώδικας SUIFrm με εντολές ABS, MIN, MAX.

3. Η ΡΟΗ ΑΝΑΛΥΣΗΣ ΕΦΑΡΜΟΓΩΝ

Όπως έχει ήδη αναφερθεί, η αρχιτεκτονική εικονικής μηχανής SUIFvm αποτελεί σημείο αναφοράς για το έργο της ροής ανάλυσης εφαρμογών. Μια διαδικασία ανάλυσης εφαρμογών αποσκοπεί στην δοκιμασία του κώδικα (επίπέδου HLL) της στοχευόμενης (για εκτέλεση σε επεξεργαστή ειδικού σκοπού) εφαρμογής ή συνόλου εφαρμογών με σημαντικές ομοιότητες πάνω στην αρχιτεκτονική (και μικροαρχιτεκτονικό πρότυπο) ενός αντιπροσωπευτικού επεξεργαστή για τον σκοπό της εξαγωγής μετρικών ώστε την διάγνωση των υπολογιστικών απαιτήσεων. Η ροή ανάλυσης εφαρμογών πρέπει να εξασφαλίζει την ανάλυση της συμπεριφοράς των εφαρμογών υπό εξέταση σε πολλαπλά επίπεδα ακρίβειας: να υπάρχει η δυνατότητα λήψης μετρήσεων κατά εκτίμηση κύκλων (cycle estimation), ακρίβειας εντολών (instruction-accurate) και ακρίβειας κύκλων (cycle-accurate). Αυτό είναι απαραίτητο καθώς τα διαφορετικά επίπεδα ακρίβειας στην προσομοίωση έχουν πολύ μεγάλες (τάξεις μεγέθους) διαφορές σε απαιτούμενο χρόνο PC επεξεργασίας και χρησιμοποιούνται στην εξαγωγή διαφορετικών χαρακτηριστικών των εφαρμογών. Για παράδειγμα, η ανάλυση πρόβλεψης διακλάδωσης (branch prediction analysis) δεν μπορεί να επιτελεστεί με εκτίμηση κύκλων, ενώ η χρήση ακρίβειας κύκλων δεν προσφέρει παραπάνω πληροφορία από την απλή λειτουργική προσομοίωση (ακρίβειας εντολών).

Όλα τα παραπάνω γίνονται εφικτά με τη χρήση κατάλληλης υποδομής μεταγλωττιστή και προσομοιωτή η οποία πρέπει να προσφέρει δυνατότητες επαναστόχευσης (προνοώντας για τις επόμενες Φάσεις σχεδιασμού ειδικών επεξεργαστών) και επέκτασης (για την οργάνωση σεναρίων ανάλυσης). Αυτές οι δυνατότητες προσφέρονται από την υποδομή μεταγλωττιστή SUIF/Machine-SUIF και την υποδομή ανάπτυξης προσομοιωτών ArchC. Οι κύριοι λόγοι που οδήγησαν στην υιοθέτηση των εργαλείων αυτών είναι:

- Προσφέρουν έτοιμες κάποιες από τις επιθυμητές δυνατότητες και επιτρέπουν την προσθήκη νέων (με τον ανάλογο ανθρώπινο κόπο). Για παράδειγμα ο Machine-SUIF είναι κατάλληλα δομημένος ώστε να επιτρέπεται η εύκολη τροποποίηση της ροής μεταγλώττισης κάτι που δεν είναι εύκολο σε άλλους μεταγλωττιστές (LCC, GCC).
- Είναι ανοικτού κώδικα ώστε να υπάρχει η δυνατότητα αποσφαλμάτωσης και επέκτασής τους.
- Ενσωματώνουν υψηλή τεχνογνωσία καθώς έχουν αναπτυχθεί από διακεκριμένες ερευνητικές ομάδες (SUIF – Stanford, Machine-SUIF – Harvard, ArchC – UNICAMP).
- Διατίθενται με μηδενικό κόστος.

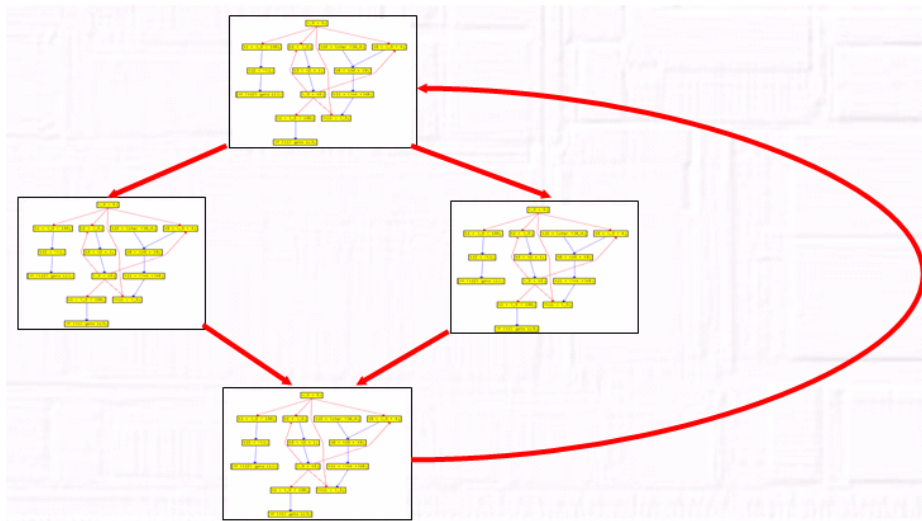
Στα αρνητικά της επιλογής αυτής (περίπου 2,5 χρόνια μετά την αρχή της ανάπτυξης της ροής ανάλυσης εφαρμογών [Kav04]) είναι ότι η ενεργός ανάπτυξη των SUIF/Machine-SUIF έχει πλήρως ανασταλεί. Σε αντιδιαστολή με αυτό, οι ερευνητές Α.Π.Θ. φρόντισαν ώστε η ροή ανάλυσης εφαρμογών (ενότητα 3.1) να μην εξαρτάται από τα συγκεκριμένα εργαλεία (SUIF2, Machine-SUIF 2, ArchC 1.6.0 και 2.0beta). Μια ανεξάρτητη ομάδα ερευνητών (όπως και ο κάθε ενδιαφερόμενος του χώρου) θα μπορούσε να ακολουθήσει την προτεινόμενη ροή χρησιμοποιώντας τα εργαλεία ανάπτυξης (μεταγλωττιστής, προσομοιωτής) της προτίμησής του. Έτσι θα μπορούσε να χρησιμοποιηθεί άλλος μεταγλωττιστής (π.χ. ο ανοικτού API/κλειστού κώδικα LANCE [LANCE] με backend συνταγμένα για τον IBURG [IBURG] ή τον OLIVE [SPAM]) και άλλη υποδομή προσομοιωτών (όπως η δημοφιλής σουίτα SimpleScalar). Η δυνατότητα αλλαγής των εργαλείων υποδομής διασφαλίζει την μακροβιότητα της ροής ανάλυσης εφαρμογών.

Στην ακόλουθη ενότητα (3.1) δίνεται αναλυτική περιγραφή της χρησιμοποιούμενης ροής ανάλυσης εφαρμογών. Σε ορισμένα σημεία επισημαίνονται πιθανές τροποποιήσεις (όπως αντικατάσταση εργαλείων) που μπορεί να προτιμηθούν από άλλους ερευνητές.

3.1. Περιγραφή της ροής ανάλυσης εφαρμογών

Η ροή ανάλυσης εφαρμογών είναι χρήσιμη σε πρώιμα στάδια του σχεδιασμού επεξεργαστών καθώς συνήθως οι μεταγλωττιστές και προσομοιωτές δεν είναι διαθέσιμοι για όλο το φάσμα των αρχιτεκτονικών λύσεων που χρειάζεται να εξεταστούν. Προκειμένου τα αποτελέσματα του χαρακτηρισμού εφαρμογών να είναι αξιοποιήσιμα για το σύνολο του πεδίου αρχιτεκτονικών επιλογών απαιτείται η χρήση μιας κοινής πλατφόρμας. Εδώ προτείνουμε τη χρήση της ενδιάμεσης αναπαράστασης μεταγλωττιστή για τον σκοπό αυτό και το εφαρμόζουμε για την περίπτωση του MachSUIF IR το οποίο είναι το σύνολο εντολών μιας γενικής RISC μηχανής που ονομάζεται SUIFvm και η οποία δεν είναι πολωμένη ως προς οποιαδήποτε υπάρχουσα μικροαρχιτεκτονική οικογένεια. Οι τεχνικές που θα περιγραφούν εδώ αλλά και στις ακόλουθες ενότητες, δεν αφορούν αποκλειστικά τα επιλεγμένα εργαλεία από τους ερευνητές Α.Π.Θ., αλλά μπορεί να γίνει αναγωγή τους και σε άλλα περιβάλλοντα ανάπτυξης που ικανοποιούν τις ίδιες αρχές. Οι βασικές αρχές είναι η χρήση IR τύπου RISC, η καταγραφή του IR σε κατάλληλα διαχειρίσιμη μορφή, η δυνατότητα λήψης μετρήσεων σε διαφορετικά επίπεδα ακρίβειας, η υποστήριξη διαφορετικών επιπέδων λεπτομέρειας στην καταγραφή των μικροαρχιτεκτονικών μοντέλων των επεξεργαστών καθώς και η επαναστοχευσιμότητα και επεκτασιμότητα των εργαλείων.

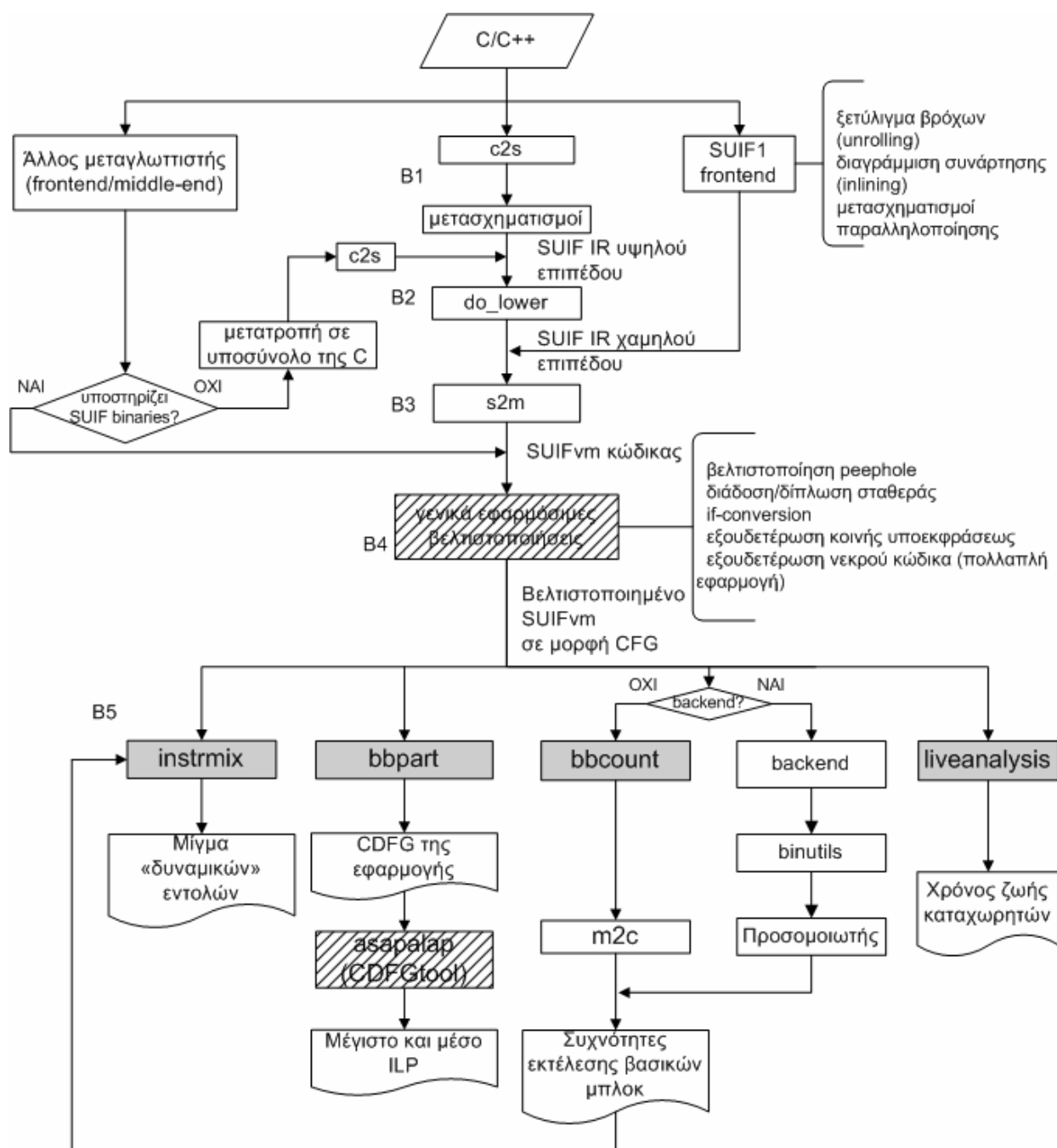
Σε κάθε περίπτωση, η εφαρμογή πρέπει να αποδομηθεί στο IR της το οποίο αποτελείται από λειτουργίες (operations) ελάχιστης πολυπλοκότητας, γνωστές επίσης ως πρωτόγονες (primitive) ή ατομικές (atomic) εντολές. Το IR αυτό είναι οργανώσιμο σε μορφή Γράφου Ροής Ελέγχου-Δεδομένων (Control Data Flow Graph - CDFG) για κάθε συνάρτηση (function/procedure), ο οποίος είναι ένας ιεραρχικός γράφος δύο επιπέδων (βλ. παράδειγμα Σχήματος 3.1) και είναι μοναδικός για κάθε συνάρτηση της εφαρμογής. Στο ανώτερο επίπεδο έχει τη μορφή Γράφου Ροής Ελέγχου (CFG) που είναι ένας κατευθυντικός γράφος έχοντας ως κόμβους τα βασικά μπλοκ και ως ακμές (edges) εξαρτήσεις ελέγχου. Στο κατώτερο επίπεδο, κάθε βασικό μπλοκ εσωκλείει έναν κατευθυντικό ακυκλικό Γράφο Εξάρτησης Δεδομένων (Data Dependence Graph – DDG) έχοντας ως κόμβους τις ατομικές εντολές και ακμές οι οποίες καταγράφουν εξαρτήσεις ορισμάτων εντολών ανάμεσα σε εντολές.



Σχήμα 3.1: CDFG για μια απλή συνάρτηση στην C. Κάθε κόμβος βασικού μπλοκ περικλείει γράφο εξάρτησης δεδομένων. Τα κόκκινα βέλη σημειώνουν εξαρτήσεις ελέγχου.

Στο Machine-SUIF, η εκτελεσιμότητα του αρχικού προγράμματος (π.χ. σε C) διατηρείται πλήρως (δηλ. δεν υπάρχει απωλεσθείσα πληροφορία) με την καταγραφή του ως SUIFvm IR. Ο αντίστοιχος SUIFvm κώδικας αποτελεί έτσι ένα είδος «εκτελέσιμης ενδιάμεσης αναπαράστασης» [Leu03] της εφαρμογής υπό ανάλυση. Ο δυναμικός χαρακτηρισμός της εφαρμογής μπορεί να γίνει σε μηχανήμα-ξενιστή (host machine) όπως ένα τυπικό PC

εκτελώντας μια C μορφή του SUIFvm IR (σύμφωνα με την ενότητα 2.4 και τον Πίνακα 2.5).



Σχήμα 3.2: Η υλοποιηθείσα ροή ανάλυσης εφαρμογών/ δοκιμασίας επιδόσεων.

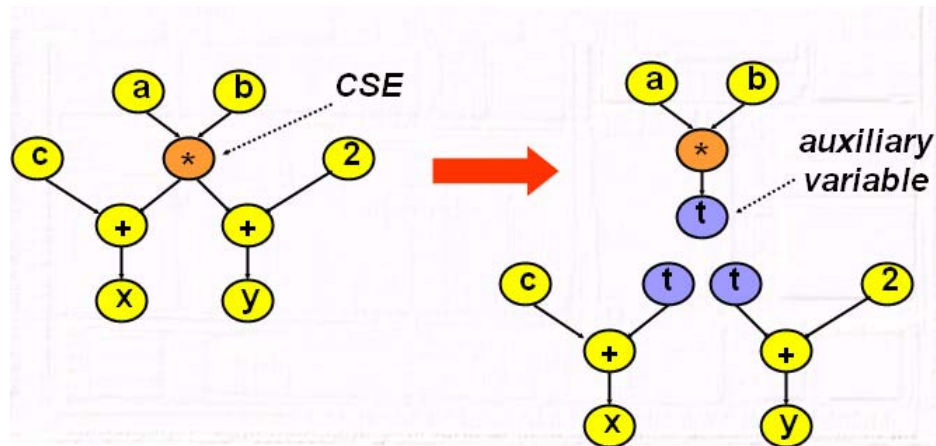
Η υλοποιηθείσα ροή ανάλυσης εφαρμογών απεικονίζεται στο Σχήμα 3.2. Τα σκιασμένα μπλοκ απεικονίζουν εργαλεία που ανέπτυξαν οι ερευνητές Α.Π.Θ., τα διαγραμμισμένα διαθέσιμα εργαλεία τα οποία τροποποιήθηκαν/επεκτάθηκαν από τους ερευνητές και τα λευκά εργαλεία που χρησιμοποιούνται στη μορφή διανομής τους. Στη ροή υπάρχει πρόβλεψη για τη δυνατότητα δοκιμασίας επιδόσεων πάνω στους ειδικούς επεξεργαστές (επιλογές με διαθέσιμο backend) που θα παραχθούν σύμφωνα με τη μεθοδολογία σχεδιασμού που θα καταγραφεί στις επόμενες Φάσεις του προγράμματος. Στις επόμενες παραγράφους της ενότητας αυτής, περιγράφονται διεξοδικά τα βήματα της διαδικασίας χαρακτηρισμού εφαρμογών κατά την ροή ανάλυσης εφαρμογών.

Στο Βήμα 1 (B1), ο C κώδικας της εφαρμογής (για C++ και Fortran υπάρχουν τα αντίστοιχα frontend) δίνεται ως είσοδος στο πέρασμα **c2s** που εκτελεί λειτουργίες του SUIF frontend. Στο στάδιο αυτό γίνεται προεπεξεργασία της εφαρμογής (με τον **cpp**) και λαμβάνεται στην έξοδο αναπαράσταση SUIF υψηλού επιπέδου (Higher SUIF). Το Higher SUIF είναι μια σύνθετη μορφή αφηρημένου συντακτικού δένδρου (AST) αποτελούμενη από απλούς (π.χ. δυαδικοί τελεστές και τελεστές ανάθεσης) και σύνθετους κόμβους (όπως δηλώσεις FOR, WHILE, SWITCH, IF-THEN-ELSE). Στο επίπεδο αυτό μπορούν να εφαρμοστούν βελτιστοποιήσεις που δεν εξαρτώνται από τη στοχευόμενη αρχιτεκτονική, όπως για διαγράμμιση συνάρτησης (function inlining), ξετύλιγμα βρόχων (loop unrolling). Μια λύση για αυτό είναι η χρησιμοποίηση περασμάτων που αναπτύχθηκαν από τρίτους για το SUIF1, αν και γενικά δε συνεργάζονται ομαλά με το MachSUIF 2.

Στο δεύτερο Βήμα (B2), εκτελείται το έγγραφο (script) **do_lower** το οποίο καθορίζει μια αλληλουχία μετασχηματισμών ανεξάρτητων από τη στοχευόμενη αρχιτεκτονική για την αποσύνθεση των σύνθετων κόμβων (όπως δηλώσεις βρόχων) σε αναπαράσταση που χρησιμοποιεί μόνο ατομικές λειτουργίες. Η παραγόμενη αναπαράσταση ονομάζεται γενικά Lower SUIF σε αντίθεση με το Higher SUIF του B1.

Στο Βήμα 3 (B3), χρησιμοποιείται το Machine-SUIF πέρασμα **s2m** για την μετατροπή από Lower SUIF σε SUIF_{vm}. Το **s2m** διαβαίνει (tree traversal) το Lower SUIF που είναι σε μορφή AST και το μετατρέπει σε SUIF_{vm} το οποίο είναι σε μορφή TAC. Παρόμοιες διεργασίες μετατροπής γίνονται στους περισσότερους μεταγλωττιστές: το AST μετατρέπεται είτε σε συλλογή από Δένδρα Ροής Δεδομένων (DFTs) ή από κατευθυντικούς ακυκλικούς γράφους (DAGs). Η διαφορά τους είναι ότι η μορφή DFT απαιτεί την εξουδετέρωση όλων των κοινών υποεκφράσεων ακόμη και αν αυτές μπορούν να οδηγήσουν σε καλύτερες συνθήκες για τη διαδικασία επιλογής κώδικα. Αντίθετα, η μορφή DAG διατηρεί τις κοινές υποεκφράσεις. Η μετατροπή από DAG σε DFT δημιουργεί πολλαπλά αντίγραφα της κοινής υποεκφράσεως ως μια νέα βοηθητική μεταβλητή (μεταβλητή “t” στο Σχήμα 3.3). Αντίστοιχα, τα εργαλεία επιλογής κώδικα χωρίζονται σε δύο βασικές κατηγορίες: αυτά που πραγματοποιούν τεχνικές τύπου tree pattern matching (π.χ. BURS) όπως τα BURG, IBURG, και OLIVE [IBURG],[SPAM] και αυτά που μπορούν να εργαστούν σε DAG όπως ο MonoBURG [MonoBURG]. Μια βασική διαφορά είναι ότι θεωρητικά τεκμηριώνεται η δυνατότητα βέλτιστης επιλογής κώδικα πάνω σε κάθε DFT ξεχωριστά [Pel88] ενώ αυτό δεν ισχύει για γενικά DAGs καθώς το πρόβλημα αποδεικνύεται ότι είναι NP-πλήρες [Ertl99]. Όπως όμως έχουμε αναφέρει ο MachSUIF δεν περιλαμβάνει εργαλείο επιλογής κώδικα κάτι που εξηγεί και τις χαμηλές επιδόσεις σε σχέση με τους LCC [LCC], GCC [GCC], και VPO [VPO],[Bou01].

Όπως φαίνεται από το Σχήμα 3.3, το DAG στο αριστερό μέρος του σχήματος θα μπορούσε να ταυτοποιηθεί ως μια ειδική εντολή με τρία ορίσματα εισόδου από μεταβλητές (a, b, c), μία σταθερά (2) και δύο ορίσματα εξόδου (x, y) από αλγόριθμο γέννησης εντολών. Η ειδική εντολή θα μπορούσε να αξιοποιηθεί κατά τη διαδικασία επιλογής εντολών μόνο στην περίπτωση που ο επιλογέας εντολών (code selector) μπορεί να εργαστεί σε DAG. Αντίθετα, η ίδια ειδική εντολή δεν είναι δυνατό να καταγραφεί ως κανόνας επιλογέα εντολών που χρησιμοποιεί τεχνικές ταιριάσματος μοτίβου δένδρου (tree pattern matching) καθώς αυτός μπορεί μόνο να διαβεί το κάθε DFT ξεχωριστά.



Σχήμα 3.3: Ένα DAG και το αντίστοιχο του δάσος από DFT. Η μετατροπή από DAG σε δάσος DFT δημιουργεί πολλαπλά αντίγραφα της κοινής υποεκφράσεως.

Κατά το B3, ο IR κώδικας δεν έχει χρονοδρομολογηθεί και δεν έχει διέλθει από φάση κατανομής καταχωρητών. Αυτή η πρώιμη μορφή του SUIFvm κώδικα αποκλείει την ύπαρξη ψευδών εξαρτήσεων στην αναπαράσταση των βασικών μπλοκ όπως αυτές που επιβάλλονται από την έγχυση καταχωρητών στη μνήμη που προκαλεί ο κατανεμητής καταχωρητών όταν δεν υπάρχουν διαθέσιμοι αδέσμευτοι καταχωρητές [Cla03].

Στη διάρκεια του Βήματος 4 (B4) εφαρμόζονται μετασχηματισμοί ανεξάρτητοι της αρχιτεκτονικής στο επίπεδο του SUIFvm όπως α) βελτιστοποίηση *reephole*, β) διάδοση σταθεράς, γ) εξουδετέρωση νεκρού (ανενεργού) κώδικα, δ) *if-conversion*, ε) κίνηση κώδικα και, εφόσον αποφασιστεί, εξουδετέρωση κοινής υποεκφράσεως.

Η βελτιστοποίηση *reephole* χρησιμοποιείται για την αντικατάσταση ακολουθιών κώδικα συμβολομεταφραστή από άλλες ισοδύναμες οι οποίες είναι καλύτερες ως προς κάποιο μετρικό, συνήθως ως προς τον αριθμό (εκτιμώμενων) κύκλων εκτέλεσης. Το MachSUIF έχει την τάση να εισάγει πλεονάζουσες εντολές αναγωγής σε τύπο (CVT) πολλές από τις οποίες μπορούν να απομακρυνθούν από τα αντίστοιχα περάσματα (**peep** και το πιο επιθετικό **peepdeux**). Επίσης, το πέραςμα CSE έχει εμβέλεια βασικού μπλοκ, ενώ αντίστοιχο πέραςμα μπορεί να εφαρμοστεί και στο επίπεδο High SUIF. Η εφαρμογή του *if-conversion* επεκτείνει το μέσο μέγεθος των βασικών μπλοκ. Για το σκοπό της γέννησης εντολών αυτό είναι γενικά ωφέλιμο αλλά όταν σκοπός είναι η μεταγλώττιση για έναν βασικό επεξεργαστή (δηλ. χωρίς ειδικές εντολές), ο απαιτούμενος χρόνος εκτέλεσης αυξάνεται καθώς μετά το *if-conversion* εκτελούνται και τα δύο εναλλακτικά μονοπάτια μιας δήλωσης *if-then-else*.

Τελικά, κατά τη διάρκεια του Βήματος 5 (B5) μπορούν να συλλεχθούν τόσο στατικά όσο και δυναμικά μετρικά για την υπό εξέταση εφαρμογή. Τα αντίστοιχα περάσματα ανάλυσης δέχονται ως είσοδο SUIFvm οργανωμένο σε CFG.

Το πέραςμα **bbpart** διαβαίνει κάθε κόμβο του CFG και κατασκευάζει τον αντίστοιχο γράφο εξάρτησης δεδομένων (DDG). Είναι δυνατή η σύνθεση ενός ιεραρχημένου CDFG έχοντας γνώση της οργάνωσης του αντίστοιχου CFG (πέραςμα **print_cfg**) η οποία και εξάγεται σε αναγνώσιμη (κειμενική) μορφή (*human readable textual form*). Το πέραςμα **bbpart** εξάγει το σύνολο των βασικών μπλοκ της εφαρμογής σε αντίστοιχη μορφή κειμένου έχοντας προσημειώσει την πληροφορία εξάρτησης δεδομένων για κάθε βασικό μπλοκ. Για την κειμενική μορφή των CFG υιοθετήθηκε η διαμόρφωση που χρησιμοποιείται από τον MachSUIF ενώ για την έξοδο του **bbpart** αναπτύχθηκε κατάλληλη διαμόρφωση. Οι EBNF γραμματικές των διαμορφώσεων αυτών καταγράφονται στην ενότητα 3.2.

Το μίγμα στατικών εντολών, δηλ. οι συχνότητες εμφάνισης των ατομικών εντολών στο SUIFvm κώδικα καταγράφονται από το πέραςμα **instrmix**. Το πέραςμα **bbcount** εισάγει μετρητές στο SUIFvm IR στην αρχή κάθε βασικού μπλοκ και στη συνέχεια ο SUIFvm

μετατρέπεται σε C με απλές αναθέσεις από το m2c. Το C πρόγραμμα που προκύπτει από αυτό χρησιμοποιείται για την καταγραφή του δυναμικού προφίλ της αναλυόμενης εφαρμογής με την εξαγωγή των συχνοτήτων εκτέλεσης των βασικών μπλοκ. Με την χρήση αυτών υπολογίζεται και το μίγμα δυναμικών εντολών της υπό εξέτασης εφαρμογής.

Το πέρασμα **liveanalysis** υπολογίζει το χρόνο ζωής καταχωρητών σημειώνοντας τους «ζωντανούς» καταχωρητές κατά την είσοδο και έξοδο από κάθε βασικό μπλοκ της εφαρμογής καλώντας αντίστοιχη ρουτίνα από τη βιβλιοθήκη **cfa** του MachSUIF. Τα αντίστοιχα αποτελέσματα βοηθούν το σχεδιαστή επεξεργαστών ειδικού σκοπού στο να αποφασίσει τον απαιτούμενο αριθμό καταχωρητών για την εξυπηρέτηση των στοχευόμενων εφαρμογών.

Ένα σημαντικό μετρικό είναι η εύρεση της εγγενούς παραλληλίας της εφαρμογής (intrinsic ILP). Η δυνατότητα αυτή δεν υφίσταται στα περισσότερα περιβάλλοντα ανάλυσης εφαρμογών με λίγες εξαιρέσεις όπως το Trimaran [Trimaran]. Στη ροή ανάλυσης εφαρμογών του προγράμματος, αυτό είναι δυνατό με τη χρήση εξωτερικών εργαλείων ανοικτού κώδικα για τη σύνθεση υψηλού επιπέδου (HLS) από CDFG. Το εργαλείο που χρησιμοποιήθηκε τελικά είναι το “CDFG toolkit” [CDFGtool]. Ο υπολογισμός της μέγιστης και μέσης παραλληλίας στα βασικά μπλοκ των εφαρμογών γίνεται με δρομολόγηση ASAP (As Soon As Possible) των εντολών-κόμβων του αντίστοιχου DDG (πέρασμα **asapalap**).

Η ροή ανάλυσης εφαρμογής είναι αξιοποιήσιμη και για την εξαγωγή χαρακτηριστικών ποσοτήτων που απαιτούν την εκτέλεση των εφαρμογών σε έναν «πραγματικό» επεξεργαστή ή/και σε μια πλήρη μικροαρχιτεκτονική. Για να είναι αυτό εφαρμόσιμο, απαιτείται η ύπαρξη αντίστοιχου backend. Μετρικά όπως αυτά που προκύπτουν από την ανάλυση κρυφής μνήμης (cache analysis) εξάγονται με την εκτέλεση του αντίστοιχου εκτελέσιμου σε λειτουργικό προσομοιωτή της αρχιτεκτονικής. Το περιβάλλον ArchC προσφέρει αρκετές δυνατότητες για την ανάπτυξη των απαιτούμενων προσομοιωτών.

Αξίζει να σημειωθεί ότι τα SUIF2 (κυρίως)/Machine-SUIF2 δεν προτείνονται «εν λευκώ» ως εργαλεία υλοποίησης μιας ροής ανάλυσης εφαρμογών καθώς ο συνδυασμός αυτός παρουσιάζει μεγάλο αριθμό μειονεκτημάτων. Είναι γνωστό, ότι το περιβάλλον SUIF1 (εκδόσεις 1.1.2 και 1.3.0.5) είναι πληρέστερο από πολλές απόψεις από το SUIF2: υποστηρίζει περισσότερες σημαντικές βελτιστοποιήσεις υψηλού επιπέδου όπως για την παραλληλοποίηση των εφαρμογών για εκτέλεση σε συμμετρικά πολυεπεξεργαστικά συστήματα μοιρασμένης μνήμης (SMP), και διανυσματοποίησης (vectorization) ενώ η βάση χρηστών έχει συνεισφέρει σε μεγαλύτερο βαθμό (περάσματα διαγράμμισης συνάρτησης, και ζετυλίσματος βρόχων).

3.2. Η χρήση της υποδομής μεταγλωττιστή στη ροή ανάλυσης εφαρμογών

Ιδιαίτερη σημασία για την εκμετάλλευση μιας υποδομής μεταγλωττιστή έχει η δυνατότητα τροποποίησης και επέκτασης της. Η υποδομή Machine-SUIF επιτρέπει την ανάπτυξη νέων περασμάτων ανάλυσης ή/και βελτιστοποίησης τα οποία μπορούν να κληθούν στο πλαίσιο μιας αλληλουχίας μεταγλώττισης (compilation sequence) κατά τον ίδιο τρόπο με τα ήδη υπάρχοντα. Επίσης σημαντική είναι η καλή διαθέσιμη τεκμηρίωση του API για τη συγγραφή νέων περασμάτων.

3.2.1. Περάσματα ανάλυσης και βελτιστοποίησης

Ο Πίνακας 3.1 συνοψίζει τα περάσματα μεταγλωττιστή που χρησιμοποιούνται από τους ερευνητές Α.Π.Θ. Στην δεύτερη στήλη δίνεται μια σύντομη περιγραφή του καθενός, στην τρίτη το είδος (ανάλυση ή/και βελτιστοποίηση) και στην τελευταία δηλώνεται η προέλευση του αντίστοιχου πηγαίου κώδικα.

Πίνακας 3.1: Περάσματα ανάλυσης ή/και βελτιστοποίησης για τον μεταγλωττιστή Machine-SUIF.

Τίτλος	Περιγραφή	Είδος	Προέλευση
bbpart	Παράγει γράφους εξάρτησης δεδομένων (DDG) για κάθε βασικό μπλοκ των συναρτήσεων της εξεταζόμενης εφαρμογής στη διαμόρφωση ISeq. Παράγει VCG αναπαράσταση των DDG [San94],[VCG].	Ανάλυσης	A.Π.Θ.
instrmix	Παράγει το μίγμα στατικών εντολών για κάθε συνάρτηση.	Ανάλυσης	A.Π.Θ.
loopstr	Καλεί τον ενσωματωμένο, στο MachSUIF, αναλυτή φυσικών βρόχων [Aho86] και είναι διαθέσιμος από την cfa.	Ανάλυσης	A.Π.Θ.
strength_reduct	Υποβιβάζει περιπτώσεις πολλαπλασιασμών και διαιρέσεων με σταθερά σε ολισθήσεις.	Βελτιστοποίησης	A.Π.Θ.
bbcount	Εισάγει μετρητές στην αρχή κάθε βασικού μπλοκ για την λήψη των συχνοτήτων εκτέλεσης βασικών μπλοκ.	Ανάλυσης (ενοργάνωση κώδικα)	A.Π.Θ.
liveanalysis	Κτίζει αλυσίδες ορισμού-χρήσης για κάθε μεταβλητή και εφαρμόζει ανάλυση χρόνου ζωής καταχωρητών βασιζόμενο στη cfa.	Ανάλυσης	A.Π.Θ.
cplx_locate	Αντικαθιστά κλήσεις σε ορισμένες αριθμητικές συναρτήσεις (abs, min, max) με χρήσεις των αντίστοιχων SUIFvm εντολών.	Βελτιστοποίησης	A.Π.Θ.
peepdeux	Βελτιστοποιητής τύπου peephole για την εξουδετέρωση των πλεοναζουσών CVT και MOV εντολών καθώς και για την απλοποίηση άλλων εκφράσεων.	Βελτιστοποίησης	A.Π.Θ.
print_cfg	Τυπώνει το CFG για κάθε συνάρτηση σε μορφή κειμένου.	Ανάλυσης	[MachSUIF]
m2c	Παράγει την αντίστοιχη C αναπαράσταση για κώδικα SUIFvm.	Εξαγωγής κώδικα	[MachSUIF]
lcse	Βελτιστοποίηση τοπικής εξουδετέρωσης κοινής υποεκφράσεως (local CSE).	Βελτιστοποίησης	[EPFL-passes]
if_conv	Βελτιστοποίηση if-conversion η οποία μπορεί να χρησιμοποιηθεί από αρχιτεκτονικές που έχουν μερική υποστήριξη δεδηλωμένης εκτέλεσης (predicated execution).	Βελτιστοποίησης	[EPFL-passes]

3.2.2. Βασική διαδικασία μεταγλώττισης από C σε SUIFvm

Για τον MachSUIF η διαδικασία της μεταγλώττισης γίνεται με την ακολουθιακή εφαρμογή περασμάτων μεταγλωττιστή (passes). Το πρώτο πέραςμα (c2s του Βήματος 1 στο Σχήμα 3.2) δέχεται C είσοδο ενώ τα υπόλοιπα εργάζονται σε δυαδική μορφή του IR της εφαρμογής. Η παρακάτω ακολουθία μεταγλώττισης (Κώδικας 3.1) επιτελεί τη διαδικασία αυτή σε πηγαίο C αρχείο ονόματι “source.c”.

```
## SUIF frontend  
  
c2s source.c source.suif  
  
## Apply frontend optimizations  
## Place to apply SUIF1 passes on source.suif.  
## Result is written to source.suif as well.
```

```
do_lower source.suif source.lsf

## Machine-SUIF passes for SUIFvm construction
## and optimization

do_s2m source.lsf source.svm
do_il2cfg source.svm source.afg

## Peephole optimizations
## Apply standard peep pass
do_peep -const_prop source.afg source.af1
do_dce source.af1 source.af1a
## Peephole optimization, part deux.
## Should be used if SUIFvm is emitted as target assembly.
do_peepdeux source.af1a source.af1b
do_dce source.af1b source.af2

## abs, min, max function replacement
do_cplx_locate source.af2 source.af3a
do_dce source.af3a source.af3
## Local CSE
do_lcse source.af3b source.af3c
do_dce source.af3c source.af3d
## Strength reduction
do_strength_reduct source.af3d source.af3e
do_dce source.af3e source.af3

## Machine-SUIF analysis and instrumentation passes

## CFG text dump
do_print_cfg -show_code source.af3
## DDG construction for each basic block
do_bbpart source.af3
## Liveness analysis
do_liveanalysis source.af3
## Natural loop analysis
do_loopstr source.af3
## Static instruction mix
do_instrmix source.af3
## Generates SUIFvm assembly
do_print source.af3 > source.asm
## Adds basic-block counters
do_bbcount source.af3 source.bbc
do_cfg2il source.bbc source.il2
## C backend emits with basic-block counters as meta-comments
do_m2c source.il2 source_m2c.c
```

Κώδικας 3.1: Παράδειγμα script για τη μεταγλώττιση από C σε SUIFvm, βελτιστοποίηση στο επίπεδο SUIFvm και εφαρμογή περασμάτων ανάλυσης. Το script χρησιμοποιείται σε περιβάλλον Linux/x86.

Ο SUIFvm κώδικας που παράγεται από το παραπάνω script έχει ορισμούς και χρήσεις τόσο προσωρινών εικονικών καταχωρητών (με πρόθεμα \$vr) όσο και ορισμάτων συναρτήσεων (arguments) οι οποίες είναι συμβολικές μεταβλητές τύπου SYM. Τα ορίσματα SYM μπορούν να αντικατασταθούν στο σύνολό τους με μετατροπή σε οργάνωση SSA και επαναμετατροπή του IR σε CFG όπως με το script του Κώδικα 3.2.

```
## Segment of a script
## Converts CFG->SSA->CFG for the purpose of eliminating all uses of
## symbolic operands (SYM_ADDR)
## *.afg is in SUIFvm CFG form

do_cfg2ssa -build_minimal_form -fold_copies -omit_useless_phi_nodes source.afg source.ssa
do_ssa2cfg source.ssa source.afgb
```

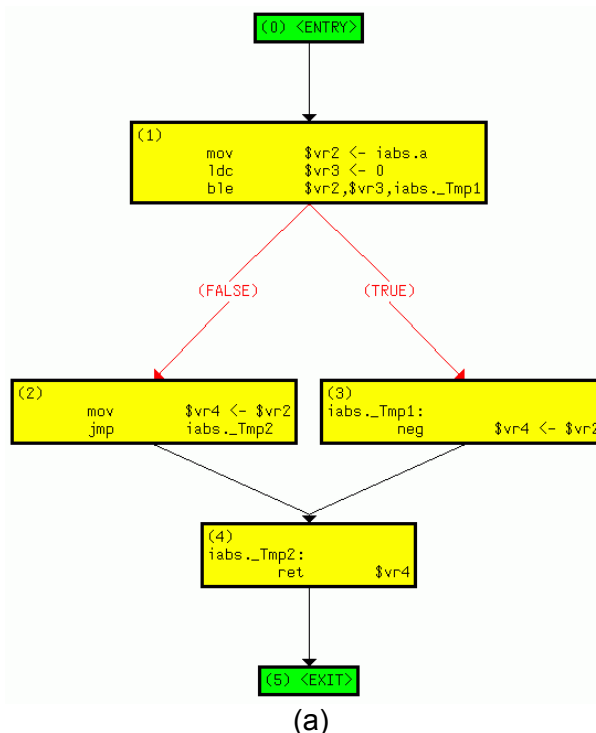
Κώδικας 3.2: Παράδειγμα script για τη μετατροπή CFG→SSA→CFG SUIFvm IR σε δυαδική μορφή. Στη νέα αναπαράσταση (source.afgb) γίνεται χρήση μόνο προσωρινών καταχωρητών και συμβολικών διευθύνσεων (ορίσματα σε εντολές LDA).

3.2.3. CDFG τυποποίηση για τις συναρτήσεις των αναλυόμενων εφαρμογών

Για την καταγραφή της πληροφορίας κάθε συνάρτησης των εφαρμογών σε CDFG αποφασίστηκε από τους ερευνητές Α.Π.Θ. αυτό να γίνεται σε επίπεδο διεπαφής αρχείου (file interface). Η κομβική αυτή απόφαση ελήφθη ώστε οι διαδικασίες ανάλυσης ροής εφαρμογών (Φάση 2) και γέννησης ειδικών εντολών και λειτουργικών μονάδων (Φάση 4) να είναι απομονώσιμες, δηλαδή διαχωρίσιμες σε επίπεδο λογισμικού και ανεξάρτητα τροποποιήσιμες. Το CDFG με τη χρήση διεπαφής αρχείου διαχωρίζει σαφώς το “frontend” (ανάλυση εφαρμογών) από το “backend” (γέννηση, επιλογή εντολών και σύνθεση λειτουργικών μονάδων). Αποτελεί δε την κοινή γλώσσα επικοινωνίας των δύο μερών του πλαισίου εργασίας σχεδιασμού ειδικών επεξεργαστών. Με τον τρόπο αυτό οι δύο ροές μπορούν να βελτιωθούν ή και να αντικατασταθούν ανεξάρτητα (εφόσον αυτό χρειαστεί μελλοντικά) χωρίς να τίθεται σε κίνδυνο η σταθερότητα του όλου πλαισίου εργασίας. Μάλιστα, τα τελευταία χρόνια η αύξηση της πολυπλοκότητας του χρησιμοποιούμενου λογισμικού έχει οδηγήσει στη γενικότερη υιοθέτηση αντίστοιχων τεχνικών για την απομόνωση των δυσκολιών όπως είναι οι μετασχηματισμοί κειμένων με εργαλεία XML.

Η καταγραφή του CDFG γίνεται από δύο αρχεία:

- Το αρχείο .cfg παράγεται από το πέρασμα print_cfg και καταγράφει το CFG μιας συνάρτησης σε επίπεδο SUIFvm. Ένα απλό παράδειγμα ενός CFG παρατίθεται στο Σχήμα 3.4 για την iabs, συνάρτησης υπολογισμού απόλυτης τιμής ακέραιου. Συγκεκριμένα στο Σχ. 3.4a δίνεται γραφική αναπαράσταση του CFG με ένθετο κώδικα SUIFvm, και στο Σχ. 3.4b το αντίστοιχο CFG σε μορφή κειμένου.
- Το αρχείο .iseq (κατά το ιστορικό όνομα «Ακολουθία Εντολών») παράγεται από το πέρασμα bbrart και καταγράφει το σύνολο των βασικών μπλοκ της συνάρτησης με τους γράφους εξάρτησης δεδομένων τους. Η αναπαράσταση iseq για την iabs δίνεται στο Σχήμα 3.4c.



```

**** CFG contains 6 blocks
**** Node # 0:      p      s 1      1 1
**** Node # 1: cbr  p 0      s 2 3      1 2
      mov    $vr2 <- iabs.a
      ldc    $vr3 <- 0
      ble    $vr2,$vr3,iabs._saveloadTmp1
**** Node # 2: ubr  p 1      s 4      1 3
      mov    $vr4 <- $vr2
      jmp    iabs._saveloadTmp2
**** Node # 3:      p 1      s 4      1 4
iabs._saveloadTmp1:
      neg    $vr4 <- $vr2
**** Node # 4: ret  p 3 2      s 5
iabs._saveloadTmp2:
      ret    $vr4
**** Node # 5:      p 4      s

```

(b)

```

Procedure name : iabs;
BB name : 0;
end bb;
BB name : 1;
iabs:l:edg1st[0][1] = $vr0.s32;
opcode=ldc;dst0=$vr0.s32;immint0=0;
opcode=ble;src0=iabs.a;src1=$vr0.s32;
end bb;
BB name : 2;
opcode=mov;dst0=iabs.temp;src0=iabs.a;
opcode=jmp;
end bb;
BB name : 3;
opcode=neg;dst0=iabs.temp;src0=iabs.a;
end bb;
BB name : 4;
opcode=ret;src0=iabs.temp;
end bb;
BB name : 5;
end bb;
end procedure;

```

(c)

Σχήμα 3.4: Μια απλή συνάρτηση (iabs) στο επίπεδο SUIFvm. (a) Γραφική αναπαράσταση του CFG με ένθετο τον κώδικα SUIFvm (η οπτικοποίηση έγινε με το πρόγραμμα VCG [VCG]). (b) Το αντίστοιχο CFG σε αναγνώσιμη μορφή. (c) Αναπαράσταση ISeq.

Η γραμματική του CFG ονομάζεται **cfginfo** και απεικονίζεται σε μορφή EBNF [EBNF] από τον Κώδικα 3.3. Αντίστοιχα, η γραμματική ISeq ονομάζεται **iseqinfo** και το EBNF της δίνεται από τον Κώδικα 3.4.

```

cfginfo_body    ::= [ cfginfo_body ] cfginfo_line
cfginfo_line    ::= cfg_entry_line
                  | bb_entry_line

cfg_entry_line  ::= <****> <cfg> <contains> T_NUM block_entry
block_entry     ::= <blocks>
bb_entry_line   ::= bb_entry_prefix dep_entry <p> pred_num_list <s> succ_num_list
bb_entry_prefix ::= <****> <node> <#> T_NUM <:>
dep_entry       ::= [ <ubr>
                  | <cbr>
                  | <mbr>
                  | <call>
                  | <ret> ]

```

```
pred_num_list ::= [ T_NUM <i>
                  | T_NUM <x>
                  | T_NUM
                  | pred_num_list T_NUM <i>
                  | pred_num_list T_NUM <x>
                  | pred_num_list T_NUM ]

succ_num_list ::= [ T_NUM <i>
                  | T_NUM <x>
                  | T_NUM
                  | succ_num_list T_NUM <i>
                  | succ_num_list T_NUM <x>
                  | succ_num_list
                  | succ_num_list T_NUM ]

T_NUM          ::= {digit}+

where:

digit matches to [0-9]
```

Κώδικας 3.3: Γραμματική EBNF για τη διαμόρφωση cfinfo.

```
iseqinfo_body      ::= { iseqinfo_line } iseqinfo_line
iseqinfo_line      ::= decoded_instr_line
                  | edglst_line
                  | bb_name_line
                  | end_bb_name_line
                  | procedure_name_line
                  | end_procedure_name_line
                  | title_line

title_line         ::= <title> ( <;> | any_name <;> )

edglst_line        ::= any_name <:> T_NUM <:> <edglst> edglst_dependency

iseq_entry         ::= <[> src_node <]> <[> dst_node <]>

edglst_dependency  ::= iseq_entry <=> any_name <;>

src_node           ::= T_NUM

dst_node           ::= T_NUM

decoded_instr_line ::= ( node_id_assign opcode_assign | opcode_assign )
                  operand_assign_list

operand_assign_list ::= [ dst_assign_list src_assign_list immint_assign_list
                  | dst_assign_list src_assign_list
                  | dst_assign_list immint_assign_list
                  | src_assign_list immint_assign_list
                  | dst_assign_list
                  | src_assign_list
                  | immint_assign_list ]

procedure_name_line ::= <procedure> <name> <:> any_name <;>

end_procedure_name_line ::= <end> <procedure> <;>

bb_name_line       ::= <bb> <name> <:> T_NUM <;>

end_bb_name_line   ::= <end> <bb> <;>

node_id_assign     ::= <node_id> <=> prefixed_num <;>

opcode_assign      ::= <opcode> <=> any_name <;>

dst_assign_list    ::= { dst_assign } dst_assign

dst_assign         ::= T_DST <=> any_name data_type_assign <;>

src_assign_list    ::= { src_assign } src_assign
```

```
src_assign      ::= T_SRC <=> any_name data_type_assign <;>
immint_assign_list ::= { immint_assign } immint_assign
immint_assign   ::= T_IMMINT <=> prefixed_num <;>
data_type_assign ::= [ T_DT_NAME <> ]
prefixed_num    ::= [ <#> ] T_NUM
any_name        ::= { anum}
T_DT_NAME       ::= <[> { anum}
T_NUM           ::= { digit}+
T_DST           ::= <dst>{ digit}+
T_SRC           ::= <src>{ digit}+
T_IMMINT        ::= <immint>{ digit}+

where:
anum matches to [a-zA-Z][a-zA-Z0-9_]*
digit matches to [0-9]
```

Κώδικας 3.4: Γραμματική EBNF για τη διαμόρφωση iseqinfo.

Για τις γραμματικές των κωδίκων 3.3 και 3.4 σημειώνουμε ότι:

- Οι τερματικές οντότητες ή απλούστερα τερματικά (terminals) εσωκλείονται σε <>.
- Οι κανόνες (rules) με πρόθεμα T_ απεικονίζουν μη-τερματικά (non-terminals).
- Τα τερματικά καθώς και λεκτικές οντότητες όπως αλφαριθμητικά (anum) και ακέραιοι βάσης 10 (digit) υποτίθεται ότι είναι αναγνωρίσιμα από τον λεκτικό αναλυτή.

Οι EBNF γραμματικές των κωδίκων 3.3 και 3.4 προέκυψαν από λεκτικούς και συντακτικούς κανόνες που συγγράφηκαν για τα γνωστά εργαλεία flex [flex] και bison [bison]. Με την ταύτιση ενός συντακτικού κανόνα επιτελείται η αντίστοιχη σημασιολογική ανάλυση όπως με την ενημέρωση εσωτερικών δομών δεδομένων για την διαχείριση της φερόμενης πληροφορίας από τα αρχεία .cfg και .iseq.

4. Η ΔΙΑΔΙΚΑΣΙΑ ΤΗΣ ΑΝΑΛΥΣΗΣ ΤΩΣ ΕΦΑΡΜΟΓΩΝ

Στην ενότητα αυτή τεκμηριώνεται η διαδικασία ανάλυσης των εφαρμογών προκειμένου την εξαγωγή χαρακτηριστικών μετρικών για αυτές. Για τις εφαρμογές του ενδιαφέροντος, τα χαρακτηριστικά αυτά διαμορφώνουν το αρχιτεκτονικό πεδίο λύσεων το οποίο προσδιορίζεται από τις γενικές παραμέτρους και ιδιότητες των στοχευόμενων επεξεργασιών ειδικού σκοπού. Η ποσοτική τους αξιολόγηση είναι απαραίτητη ώστε να επισημανθούν οι καλύτερες από τις λύσεις αυτές κατά την Φάση 3. Επειδή το πεδίο λύσεων είναι πολυδιάστατο (με συνιστώσες τις διαφορετικές παραμέτρους/ιδιότητες), δεν αναμένεται να υφίσταται μία μοναδική λύση σε κάθε πρόβλημα, αλλά ομάδες μερικά βέλτιστων λύσεων, διευθετημένες σε αντίστοιχες καμπύλες Pareto [Gri03].

Στην ενότητα 4.1 καταγράφεται το σύνολο των χαρακτηριστικών που οφείλεται να εκτιμηθούν και τεκμηριώνεται ο τρόπος με τον οποίο αυτό θα επιτελεστεί. Στην ενότητα 4.2 παρουσιάζονται οι εφαρμογές που επιλέχθηκαν προς ανάλυση: πρόκειται για εφαρμογές-διαπυρήνες που απαντώνται με μεγάλη συχνότητα σε συγκεκριμένα πεδία χρήσης (όπως επεξεργασία εικόνας και video, πρωτόκολλα διαχείρισης δικτύων και επικοινωνιών). Τέλος, στην ενότητα 4.3 παρουσιάζονται οι μετρήσεις για τον χαρακτηρισμό των εφαρμογών, και εκτιμώνται τα αντίστοιχα μετρικά. Η πληροφορία αυτή θα καθοδηγήσει τον καθορισμό των

αρχιτεκτονικών και μικροαρχιτεκτονικών προτύπων για τους επεξεργαστές ειδικού σκοπού ανά εφαρμογή ή πεδίο εφαρμογών.

4.1. Μετρικά χαρακτηρισμού εφαρμογών και διαδικασία εξαγωγής τους

Η προτεινόμενη ροή ανάλυσης των εφαρμογών μπορεί να χρησιμοποιηθεί για την εξαγωγή χαρακτηριστικών των εφαρμογών που είναι χρήσιμα στο σχεδιασμό των ASIP. Η διαδικασία μεταγλώττισης (από C σε κώδικα SUIFvm) επιδέχεται μία σειρά παραμέτρων για τη δημιουργία διαφορετικών σημείων λύσεων (π.χ. με διαφορετικές επιδόσεις σε κύκλους εκτέλεσης, στατικό μέγεθος κώδικα συμβολομεταφραστή, κ.λ.π.). Σε υψηλό επίπεδο λαμβάνονται αποφάσεις όπως για την έκταση της εφαρμογής ξετυλίγματος βρόχων. Στον Πίνακα 4.1 δίνονται αναλυτικά οι παράμετροι που μπορούν να τεθούν κατά τη διαδικασία της μεταγλώττισης.

Πίνακας 4.1: Παράμετροι της διαδικασίας μεταγλώττισης από τον MachSUIF προκειμένου την εξαγωγή στατικών και δυναμικών χαρακτηριστικών.

Παράμετρος	Περιγραφή	Άμεση εξάρτηση από λογισμικό	Επίπεδο εφαρμογής
IFC	Επίδραση της εφαρμογής μετασχηματισμού if-conversion.	if_conv	MachSUIF
CSE	Εξουδετέρωση κοινής υποεκφράσεως τόσο μέσω SUIF2 (με το pass JuanCSE [EXPRESS]) όσο και μέσω Machine-SUIF.	JuanCSE [EXPRESS], lcse	SUIF2, MachSUIF
MOREPEEP	Εξουδετέρωση πλεοναζουσών εντολών ανάθεσης τύπου	peepdeux	MachSUIF
CFGSSACFG	Μετατροπή CFG→SSA→CFG για την ανάθεση όλων των μεταβλητών σε εικονικούς καταχωρητές.	ssa, cfg2ssa, ssa2cfg	MachSUIF
INL	Διαγράμμιση συνάρτησης.	JuanInlining	SUIF2
UNL	Ξετύλιγμα βρόχων σε ρυθμιζόμενη έκταση.	JuanLoop ή με επεξεργασία πηγαίου κώδικα	SUIF2

Για κάθε διαφορετικό σημείο του «πεδίου των μεταγλωττίσεων» (compilation space) το SUIFvm IR καταγράφεται σε μορφή ISeq και παράγεται πλήρως αυτόματα το αντίστοιχο C backend αρχείο για την εξαγωγή των δυναμικών χαρακτηριστικών. Ο Πίνακας 4.2 συνοψίζει τη διαδικασία εξαγωγής στατικών και δυναμικών χαρακτηριστικών από τις εφαρμογές. Για τα περισσότερα από αυτά (όλα πλην του LIFE) παραθέτονται αναλυτικές μετρήσεις στην ενότητα 4.3.

Πίνακας 4.2: Χαρακτηριστικές ποσότητες που εξάγονται κατά την ανάλυση των εφαρμογών.

Ποσότητα (μία τιμή ή σειρά τιμών)	Περιγραφή	Άμεση εξάρτηση από λογισμικό	Είδος χαρακτηριστικού
CDFG σε μορφή ISeq και cdfg [CDFGtool]	Καταγραφή του CDFG για κάθε συνάρτηση σε κάθε εφαρμογή. Τα CDFG φέρουν επιπλέον πληροφορία τύπων δεδομένων για τα ορίσματα των εντολών.	bbpart, αναλυτής εφαρμογών A.Π.Θ., iseqinfo, cfginfo	Στατικό (CDFG) και στατικό/δυναμικό (κατανομή τύπων δεδομένων)
LIFE	Χρόνος ζωής καταχωρητών.	lifeanalysis	Στατικό
INUM	Μέγεθος βασικών μπλοκ και συναρτήσεων.	bbpart	Στατικό
BBC	Αριθμός βασικών μπλοκ.	bbcoun	Στατικό
SIMIX	Μίγμα στατικών εντολών.	instrmix	Στατικό
FREQ	Συχνότητες εκτέλεσης βασικών μπλοκ.	bbcoun, m2c	Δυναμικό
DIMIX	Μίγμα δυναμικών εντολών.	instrmix, FREQ	Δυναμικό
MAXILP, AVGILP	Μέγιστο και μέσο εγγενές ILP (παράλληλια απλών εντολών) για κάθε βασικό μπλοκ.	asapalap, CDFGtool, cdfg	Στατικό (στατική χρονοδρομολόγηση)

4.2. Εφαρμογές δοκιμής

Η υλοποιηθείσα ροή ανάλυσης εφαρμογών του Σχήματος 3.2 χρησιμοποιήθηκε σε αντιπροσωπευτική ομάδα 9 εφαρμογών (Πίνακας 4.3), οι οποίες εμφανίζονται συχνά σε διαδικασίες δοκιμής επιδόσεων. Οι εφαρμογές έχουν συλλεχθεί από σουίτα εφαρμογών πυρήνων (kernel benchmarks) που αναπτύχθηκε για τις μετρήσεις που αναφέρονται σε εργασία που έχει υποβληθεί στο IEEE Transactions on Computers. Η σουίτα ονομάζεται ZOLC-bench και περιλαμβάνει μεταξύ άλλων τις εφαρμογές crc, edgedet, fsme, fsp6c1, kmprskip, lcs, matmult, mc, tssme. Οι πυρήνες αυτοί απαντώνται σε εφαρμογές επεξεργασίας εικόνας, βίντεο και ταύτισης/ανάλυσης σειρών δεδομένων.

Πίνακας 4.3: Σύνοψη των εφαρμογών δοκιμής για τη ροή ανάλυσης εφαρμογών.

Εφαρμογή	Περιγραφή	Αναφορά
crc	Cyclic redundancy check	--
fsme	Full-search motion estimation	--
fs6c1	Full-search motion estimation after data-reuse (source-level) transformations	--
tssme	Three-step search block-matching motion estimation	[Koga81]
matmult	Blocked matrix multiplication	--
edgedet	Gradient-based edge detection	
mc	Motion compensation	
kmprskip	Knuth-Morris-Pratt string matching algorithm	[ChaLec]
lcs	Least common substring	[ChaLec]

4.3. Πειραματικές μετρήσεις και χαρακτηρισμός των εφαρμογών

4.3.1. Επίδραση των βελτιστοποιήσεων μεταγλωττιστή στις επιδόσεις των εφαρμογών

Οι δυνατότητες του χρησιμοποιούμενου μεταγλωττιστή επηρεάζουν σημαντικά τις επιδόσεις της εξεταζόμενης εφαρμογής (μέγεθος κώδικα, αριθμός δυναμικών εντολών) σε συνδυασμό με την αρχιτεκτονική του επεξεργαστικού συστήματος στο οποίο εκτελείται. Για παράδειγμα, σε έναν επεξεργαστή με δυνατότητα ταυτόχρονης έκδοσης μίας εντολής, η επίδραση του ξετυλίγματος βρόχων είναι αμελητέα, καθώς δεν μπορεί να εκμεταλλευτεί την αναδεικνυόμενη παραλληλία της εφαρμογής. Ένας n -way VLIW επεξεργαστής μπορεί να εκδώσει μέχρι n εντολές (για την ακρίβεια: παράλληλες μικρολειτουργίες) ταυτόχρονα και με έναν καλό μεταγλωττιστή μπορεί να αποδώσει ένα ILP που να πλησιάζει σε ορισμένες περιπτώσεις το n .

Στον Πίνακα 4.4 δίνονται τα γενικά χαρακτηριστικά εφαρμογών από τον Πίνακα 4.3. Ο μεταγλωττιστής είναι ο MachSUIF στοχεύοντας SUIFvm κώδικα. Ως βασική βελτιστοποίηση αναφέρεται η χρήση των περασμάτων: peer, peerdeux, cplx_locate, strength_reduct, και πολλαπλή χρήση του dce. Πρόσθετη βελτιστοποίηση κωδικοποιείται ως {IFC, CSE+LCSE, CFGSSACFG, INL, UNR}, με τα ψηφία 1,0 να υποδεικνύουν τη χρήση ή μη, αντίστοιχα, των παραμέτρων αυτών. Αριθμητική τιμή διάφορη του 0 για το UNL υποδεικνύει την έκταση του ξετυλίγματος βρόχων. Οι στήλες 3-6 δίνουν τα αναγραφόμενα χαρακτηριστικά.

Πίνακας 4.4: Γενικά χαρακτηριστικά των εφαρμογών δοκιμής για μεταγλώττιση με MachSUIF. Για τις εφαρμογές (πλην των kmprskip, lcs) δίνονται αποτελέσματα για τη βασική βελτιστοποίηση και την καλύτερη από αυτές που εφαρμόζουν ξετύλιγμα βρόχων.

Εφαρμογή	Βελτιστοποίηση	Αριθμός δυναμικών εντολών	Αριθμός στατικών εντολών	Αριθμός BB	Εντολές ανά BB
crc	Βασική	33764	213	29	7.34
crc	00008	33031	647	59	10.97
edgedet	Καμία	129015	59	17	3.47
edgedet	11018	116939	401	49	8.18
fsme	Καμία	34008026	107	36	2.97
fsme	01018	33265504	524	125	4.19
fsp6c1	Καμία	18990090	305	92	3.32
fsp6c1	11018	16358078	941	139	6.77
kmpskip	Καμία	32070	334	93	3.59
kmpskip	11018	25045	305	80	3.81
lcs	Καμία	1637984	140	34	4.12
lcs	11018	1409918	622	82	7.59
matmult	Καμία	870113	71	18	3.94
matmult	01008	751329	224	18	12.44
mc	Καμία	100062	177	24	7.38
mc	01008	71709	1222	87	14.05
tssme	Καμία	4420183	184	44	4.18
tssme	01008	4362239	916	316	2.90

Από τον Πίνακα 4.4 διαπιστώνουμε την θετική επίδραση του ξετυλίγματος βρόχων στον μέσο αριθμό εντολών ανά βασικό μπλοκ (BB). Η αύξηση του αριθμού των BB συμβαίνει όταν υπάρχουν αλλαγές ροής προγράμματος στο τμήμα κώδικα που εφαρμόζεται το ξετύλιγμα βρόχων (UNL).

Προκειμένου να έχουμε περισσότερο ολοκληρωμένη εικόνα για την επίδραση του μεταγλωττιστή (ήτοι της ποιότητάς του) οι ίδιες εφαρμογές μεταγλωττίστηκαν με gcc-3.4.3 [GCC] για την αρχιτεκτονική DLX. Το DLX backend δεν βασίζεται σε κάποιο από τα γνωστά backend του DLX (υπάρχουν 2: για τον gcc-1.39 και 2.7.2.3): συντάχθηκε από ερευνητή Α.Π.Θ. (Ν. Καββαδίας), και είναι σε πρώτη κατάσταση (alpha version) καθώς το έργο επαναστόχευσης του gcc είναι χρονοβόρο και απαιτητικό. Στο DLX backend έχουν συμπεριληφθεί ορισμένες αρχιτεκτονικές επεκτάσεις όπως η εντολή SELECT. Οι παρακάτω μετρήσεις για τον DLX (Πίνακας 4.5) δεν επιδέχονται άμεσης σύγκρισης με τον SUIFvm, όπως αυτό δεν μπορεί να γίνει για τους MachSUIF και GCC. Αυτό οφείλεται στο ότι ο SUIFvm δεν χρησιμοποιεί στοίβα ορισμάτων (argument stack) για την διαχείριση της οποίας στον DLX (και σε κάθε κλασσικό RISC) απαιτούνται δαπανηρές ακολουθίες φόρτωσης και αποθήκευσης για ορίσματα και σωζόμενους καταχωρητές, και ενημέρωση του δείκτη στοίβας (SP: Stack Pointer), της διεύθυνσης επιστροφής (RA: Return Address) και ενδεχόμενα του δείκτη πλαισίου (FP: Frame Pointer). Επίσης για τον SUIFvm έχει ήδη γίνει ανάθεση καταχωρητών (κατανομή καταχωρητών με άπειρους διαθέσιμους καταχωρητών και χωρίς fill/spill στη μνήμη) που επίσης ευνοεί τις επιδόσεις του MachSUIF. Για τον GCC χρησιμοποιούνται δύο διαφορετικές βελτιστοποιήσεις: **-O0** (όπου δεν γίνεται κατανομή καταχωρητών αλλά και καμία βελτιστοποίηση) και υψηλή: **-O3 -fno-optimize-sibling-calls -mhw-select** (εφαρμογή των βελτιστοποιήσεων και χρήση εντολών SELECT). Η προσομοίωση ακρίβειας εντολής έγινε για μοντέλο του DLX σε ArchC.

Πίνακας 4.5: Η επίδραση των βελτιστοποιήσεων στον GCC για τις εφαρμογές δοκιμής.

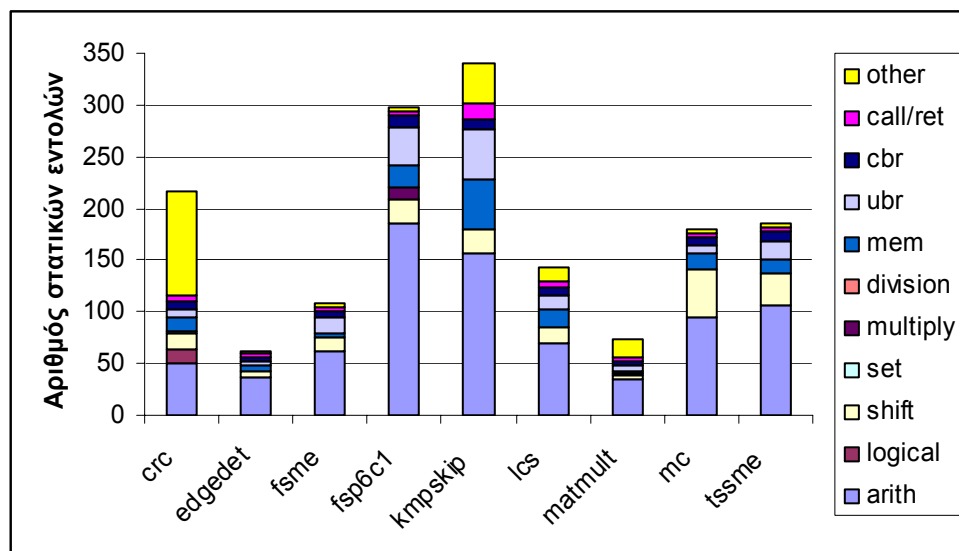
Εφαρμογή	Βελτιστοποίηση	# Instrs.	% μείωση εντολών	επιτάχυνση λόγω μεταγλωττιστή
----------	----------------	-----------	------------------	-------------------------------

crc	-O0	53106	47.31	1.90
crc	υψηλή	27979		
edgedet	-O0	304383	70.10	3.34
edgedet	υψηλή	91003		
fsme	-O0	92185270	78.68	4.69
fsme	υψηλή	19652004		
fsp6c1	-O0	45464306	69.46	3.27
fsp6c1	υψηλή	13886096		
kmpskip	-O0	93949	64.86	2.85
kmpskip	υψηλή	33011		
lcs	-O0	7356102	85.86	7.07
lcs	υψηλή	1040268		
matmult	-O0	1203659	71.07	3.46
matmult	υψηλή	348170		
mc	-O0	602333	82.41	5.68
mc	υψηλή	105969		
tssme	-O0	12777151	84.99	6.66
tssme	υψηλή	1917263		

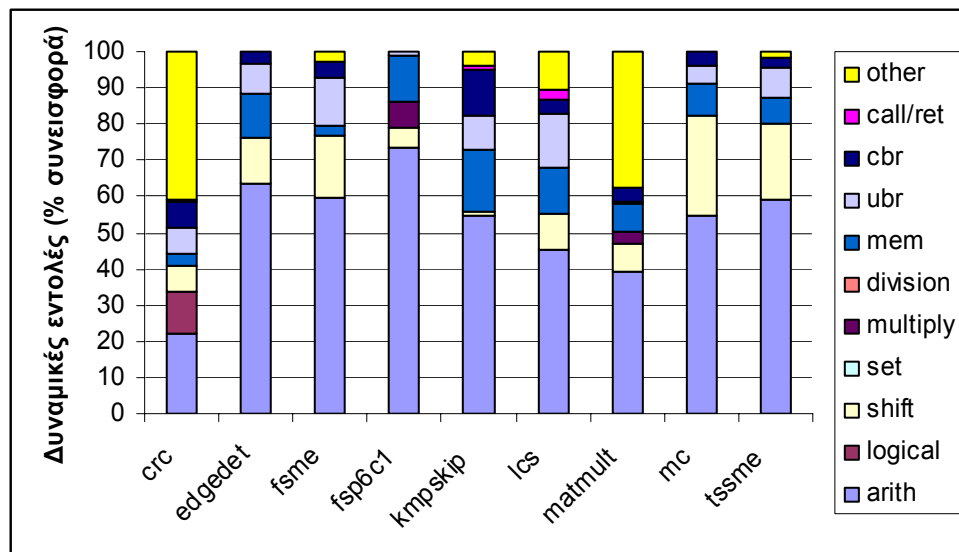
Η μέση επιτάχυνση των εφαρμογών δοκιμής είναι 4.33 λόγω της εφαρμογής των βελτιστοποιήσεων της GCC. Μπορούμε να πούμε ότι το αποτέλεσμα αυτό είναι αναμενόμενο για ένα μεταγλωττιστή με ισχυρές βελτιστοποιήσεις (δηλαδή μια κατά εκτίμηση επιτάχυνση της εφαρμογής της τάξης του 4*) συμφωνεί με τον εμπειρικό νόμο του Proebstring [Proebstring]. Για τον ίδιο λόγο η έλλειψη ισχυρών βελτιστοποιήσεων εξηγεί την αντίστοιχη χαμηλή επιτάχυνση (1.15) για το πλαίσιο SUIF/MachSUIF.

4.3.2. Μίγμα στατικών εντολών (SIMIX) και μίγμα δυναμικών εντολών (DIMIX)

Για την εξαγωγή των SIMIX και DIMIX χρησιμοποιείται η κατηγοριοποίηση των εντολών στα διαφορετικά είδη του Πίνακα 2.2 με τους εναλλακτικούς συμβολισμούς (στήλη 3). Στο Σχήμα 4.1 δίνεται το μίγμα στατικών εντολών για την περίπτωση βασικής βελτιστοποίησης με MachSUIF. Στο Σχήμα 4.2 δίνεται αντίστοιχα το μίγμα δυναμικών εντολών για τον υπολογισμό του οποίου χρησιμοποιήθηκαν οι συχνότητες βασικών μπλοκ για κάθε εφαρμογή.



Σχήμα 4.1: Μίγμα στατικών εντολών για τις εφαρμογές δοκιμής.



Σχήμα 4.2: Μίγμα δυναμικών εντολών για τις εφαρμογές δοκιμής.

Συνοψίζοντας, από τα αποτελέσματα του Σχήματος 4.2 προκύπτει ότι στο μίγμα δυναμικών εντολών κυριαρχούν οι αριθμητικές (arith) εντολές σχεδόν για όλες τις εφαρμογές (πλην της crc) με κατά μέσο όρο συνεισφορά 52.5%. Κατά δεύτερο λόγο απαντώνται εντολές ολίσθησης (12.15%), διακλάδωσης (αθροιστική συνεισφορά από ubr και cbr :12.12%) και προσπέλασης στη μνήμη δεδομένων (9.4%). Το σημαντικό ποσοστό των «άλλων» (other) εντολών (10.85%) οφείλεται στη μεγάλη συνεισφορά από εντολές ανάθεσης τύπου (CVT). Το ποσοστό των «άλλων» εντολών γενικά περιορίζεται με τη χρήση του περάσματος reepdeux, παρουσιάζεται δε αυξημένο για τις crc και matmult καθώς για αυτές το πέρασμα αυτό έπρεπε να απενεργοποιηθεί λόγω σφαλμάτων στη μεταγλώττιση. Η χρήση του reepdeux δεν είναι απαραίτητη όταν γίνεται επιλογή κώδικα (do_gen) και κατανομή καταχωρητών (do_raga) για κάποια πραγματική αρχιτεκτονική. Το γεγονός αυτό έχει επιβεβαιωθεί από τους ερευνητές Α.Π.Θ. για το πειραματικό SUIFm backend που έχει αναφερθεί στην ενότητα 3.

4.3.3. Ανάλυση τύπων δεδομένων

Τα αρχεία iseqinfo που καταγράφουν τους γράφους εξάρτησης δεδομένων για τα BB κάθε εφαρμογής μπορούν να φέρουν και πληροφορία τύπων δεδομένων (ΤΔ) για τα ορίσματα (έντελα) SUIFm εντολών. Οι επιτρεπόμενοι τύποι είναι αυτοί του Πίνακα 2.4. Η ρητή αναφορά του τύπου καταγράφεται ως ο συντακτικός κανόνας data_type_assign της EBNF σύνταξης για τη διαμόρφωση iseqinfo. Οι ρητές αναφορές τύπων στα αρχεία iseqinfo των εφαρμογών συλλέγονται από αυτόματο λογισμικό εργαλείο και παράγονται στατιστικά κατά επιλογές. Οι διαθέσιμες επιλογές είναι του τύπου -dt-mode-clustering στην οποία συμπεριλαμβάνονται οι παρακάτω:

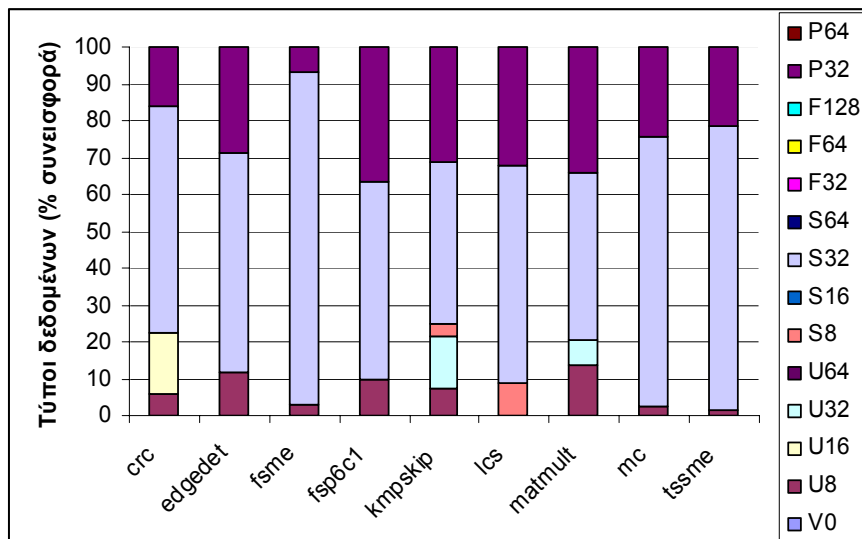
- -dt-static-proc: μέτρηση εμφανίσεων τύπων δεδομένων σε στατικές εντολές ανά συνάρτηση
- -dt-static-bb: όπως παραπάνω σε στατικές εντολές ανά BB
- -dt-dynamic-proc: όπως παραπάνω σε δυναμικές εντολές ανά συνάρτηση
- -dt-dynamic-bb: όπως παραπάνω σε δυναμικές εντολές ανά BB

Από τις τέσσερις προαναφερθείσες εκδοχές των στατιστικών των ΤΔ, βασικότερη είναι η απεικόνιση `dt-dynamic-proc`, που καταγράφει τις χρήσεις ορισμάτων εντολών (άρα και των τύπων τους) σε συνθήκες εκτέλεσης του προγράμματος. Η πληροφορία που συλλέγεται έτσι χρησιμοποιείται για να αποφασιστούν τα εξής:

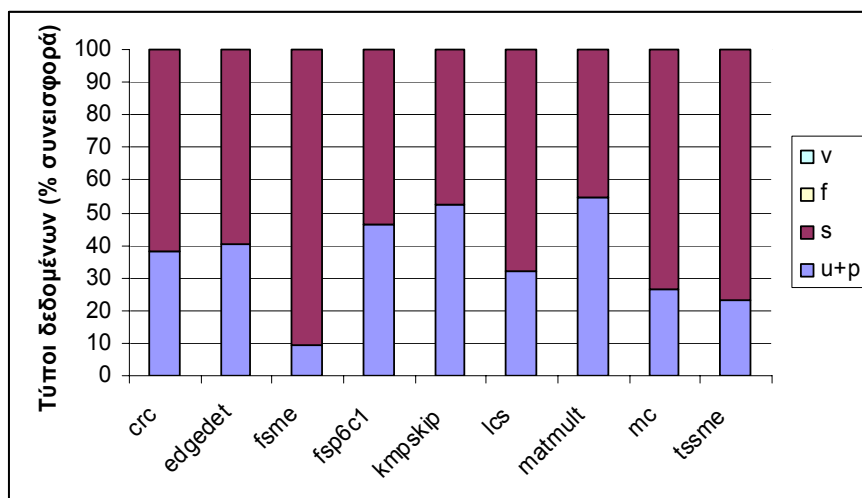
- την απαίτηση για λειτουργικές μονάδες ακεραίων ή/και αριθμών κινητής υποδιαστολής
- το εύρος bit του διαδρόμου δεδομένων

Σε μεταγενέστερη φάση όταν οι ειδικές εντολές είναι πλέον γνωστές (αντικείμενο της Φάσης 4) είναι αναγκαίο να εξάγεται πληροφορία ΤΔ ανά ειδική εντολή ή ομάδα ειδικών εντολών που χαρτογραφείται στην ίδια λειτουργική μονάδα. Από αυτή καθορίζεται το εύρος bit της λειτουργικής μονάδας και οι απαιτήσεις σε μονάδες αναγωγής εύρους bit (extension ή truncation) στο υλικό.

Στο Σχήμα 4.3 δίνονται αποτελέσματα ανάλυσης ΤΔ με τις συνολικές εμφανίσεις τύπων δεδομένων ανά εφαρμογή. Υπολογίζονται με άθροιση των επι μέρους εμφανίσεων ανά συνάρτηση (`-dt-dynamic-proc`).



(a)



(b)

Σχήμα 4.3: % εμφανίσεις τύπων δεδομένων σε δυναμικές εντολές ανά εφαρμογή. (α) Ανά ΤΔ κατά τον Πίνακα 2.4. (β) Με ομαδοποίηση σε απρόσημους (u+p) και προσημασμένους ακέραιους (s), κινητής υποδιαστολής (f) και άλλο (v).

Πρέπει να σημειωθεί εδώ ότι η επάρκεια της ανάλυσης τύπων δεδομένων εξαρτάται από τη χρησιμοποίηση κατάλληλων τύπων για κάθε μεταβλητή στο επίπεδο HLL (γλώσσα C).

4.3.4. Εξαγωγή και χαρακτηρισμός των κρίσιμων βασικών μπλοκ των εφαρμογών

Για την εξαγωγή των βασικών μπλοκ της εφαρμογής, έγινε κατάταξή τους (ranking) με αυτόματο εργαλείο, σύμφωνα με τον αριθμό δυναμικών εντολών (αριθμός στατικών εντολών × συχνότητα εκτέλεσης BB). Στον Πίνακα 4.6 δίνονται: η αρίθμηση του BB ως συνδυασμός του αύξοντα αριθμού συνάρτησης και της θέσης του σε αυτή (#PROC-BB), ο αριθμός εντολών (# κμβ), και η % συνεισφορά του στο συνολικό αριθμό δυναμικών εντολών της αντίστοιχης εφαρμογής. Αυτό γίνεται για δύο διαφορετικούς συνδυασμούς βελτιστοποίησης μεταγλωττιστή (εκτός για τις εφαρμογές kmprskip και lcs για τις οποίες δίνεται η κωδικοποίηση 00000) κατά την ορολογία του Πίνακα 4.4.

Πίνακας 4.6: Γενικά χαρακτηριστικά των «κρίσιμων» (hot) βασικών μπλοκ των εφαρμογών δοκιμής. Απεικονίζονται ως και τα 5 σημαντικότερα BB για κάθε εφαρμογή.

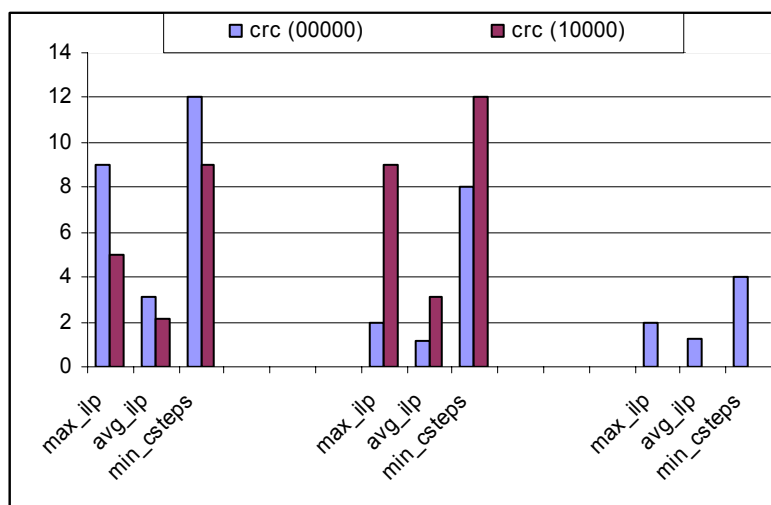
Εφαρμογή	Κωδικ.	# PROC-BB	# κμβ	% εντολών	# PROC-BB	# κμβ	% εντολών	# PROC-BB	# κμβ	% εντολών	# PROC-BB	# κμβ	% εντολών	# PROC-BB	# κμβ	% εντολών	% συν. αριθμός εντολών
crc	00000	1-5	36	27.30	0-4	8	24.26	0-3	4	24.26							75.82
crc	10000	0-3	18	72.04	1-5	36	18.01										90.05
edgedet	00000	1-5	14	44.45	1-7	13	34.37	1-11	7	6.11							84.93
edgedet	11008	1-5	15	6.57	1-31	16	6.03	1-21	16	5.99	1-11	16	5.99	1-36	16	5.91	30.49
fsme	00000	1-13	16	43.36	1-14	5	13.39	1-17	5	8.31	1-20	5	8.22	1-26	3	8.13	81.41
fsme	10000	1-13	16	39.02	1-14	5	12.06	1-16	5	11.97	1-22	5	11.88	1-18	5	11.88	86.81
fsp6c1	00000	1-81	14	67.94	1-82	3	14.56	1-67	13	9.45							91.95
fsp6c1	11008	1-129	115	80.99	1-115	43	9.89										90.88
kmpskip	00000	0-3	5	64.36	3-2	8	6.44	4-31	14	5.46							76.26
lcs	00000	1-5	6	24.01	1-6	5	20.01	1-11	17	18.65	1-13	6	17.42	1-14	2	5.81	85.90
matmult	00000	1-11	26	97.91													97.91
matmult	01008	1-11	179	97.58													97.58
mc	00000	1-9	11	45.03	1-14	20	30.70	1-16	20	19.19							94.92
mc	01008	1-9	11	7.85	1-77	20	5.35	1-68	20	5.35	1-59	20	5.35	1-50	20	5.35	29.25
tssme	00000	1-21	23	38.36	1-22	12	19.99	1-25	12	12.48	1-28	12	12.46				83.29
tssme	10000	1-21	23	33.16	1-22	12	17.30	1-24	12	17.26	1-26	12	17.21	1-30	5	7.17	92.10

Παρατηρώντας την τελευταία στήλη του Πίνακα 4.6 (% συνολικός αριθμός εντολών) διαπιστώνουμε ότι κατά μέσο όρο, τα 5 BB με την υψηλότερη κατάταξη καλύπτουν το 80.6% του συνολικού αριθμού εντολών των εφαρμογών. Η επίδραση του loop unrolling όταν υφίστανται αλλαγές ροής ελέγχου εντός της περιοχής κώδικα όπου εφαρμόζεται έγκειται στη κατανομή των BB υψηλής βαρύτητας σε μεγαλύτερο αριθμό BB με αναλογικά χαμηλότερη βαρύτητα. Αυτό δικαιολογεί τη μείωση του βάρους σε εφαρμογές όπως ο `edgedet` (11008) όπου το βάρος μειώνεται από 84.9% σε 30.5%. Αντίθετα σε περιπτώσεις όπου δεν υπάρχουν αλλαγές ροής ελέγχου σε εσωτερικό βρόχου όπου εφαρμόζεται loop unrolling, ο αριθμός εντολών (# κμβ) ανά BB αυξάνεται σημαντικά. Αυτό συμβαίνει π.χ. στην περίπτωση του `fsrbc1` (11008) όπου ο αριθμός εντολών αυξάνεται από 14 σε 115.

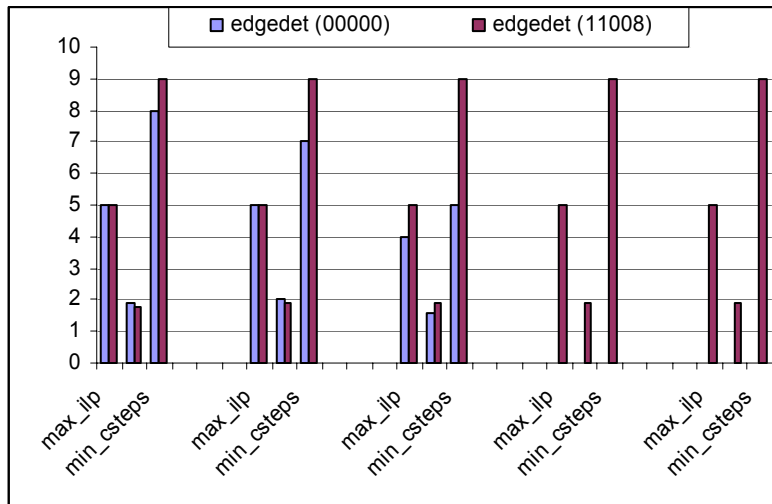
Στη συνέχεια εξήχθησαν αρχεία `iseqinfo` για κάθε βασικό μπλοκ ξεχωριστά και έγινε αυτόματη μετάφραση (κατά το BNF της διαμόρφωσης CDFG [CDFGtool-guide]) σε αναπαράσταση CDFG. Τα CDFG αυτά χρονοδρομολογήθηκαν κατά ASAP με εργαλείο (`asapalar` που αναφέρεται στην ενότητα 3.1) που χρησιμοποιεί το API του CDFG toolset. Για κάθε BB μπορούν να υπολογιστούν:

- **max_ilo**: η μέγιστη παραλληλία εντολών SUIFvm σε ένα βήμα ελέγχου (control step)
- **min_csteps**: ο (ελάχιστος) αριθμός των απαιτούμενων βημάτων ελέγχου για την χρονοδρομολόγηση του DFG κάθε BB
- **avg_ilo**: η μέση παραλληλία εντολών, η οποία προκύπτει από την έκφραση $\text{min_csteps}/\text{max_ilo}$

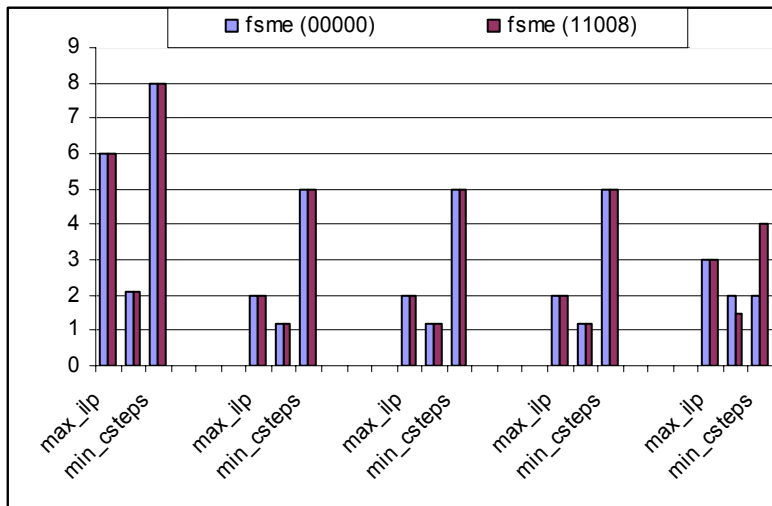
Στο Σχήμα 4.5 (α ως ι) δίνονται αναλυτικά οι τρεις ποσότητες για δύο διαφορετικούς συνδυασμούς βελτιστοποίησης μεταγλωττιστή κατά την ορολογία του Πίνακα 4.4.



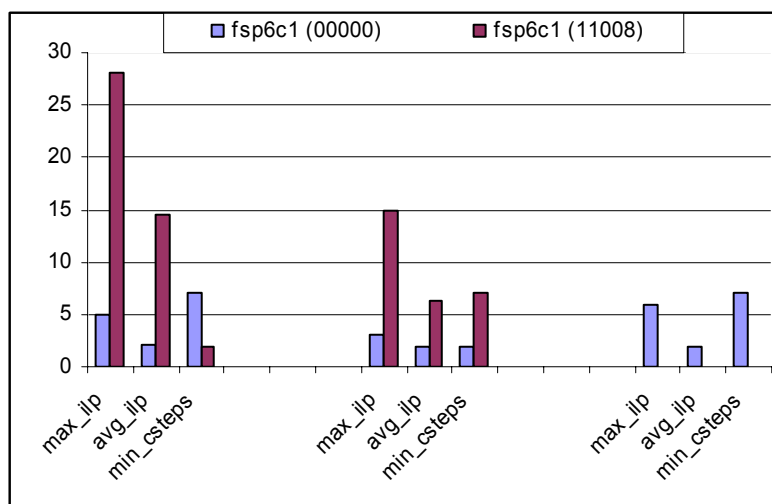
(a)



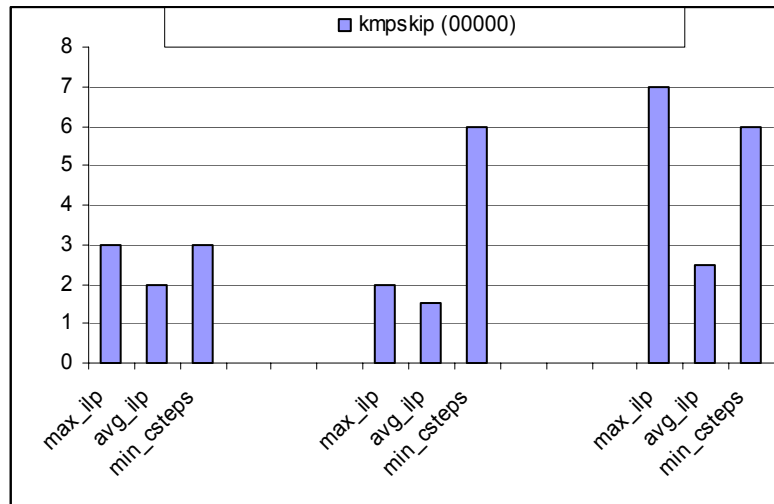
(b)



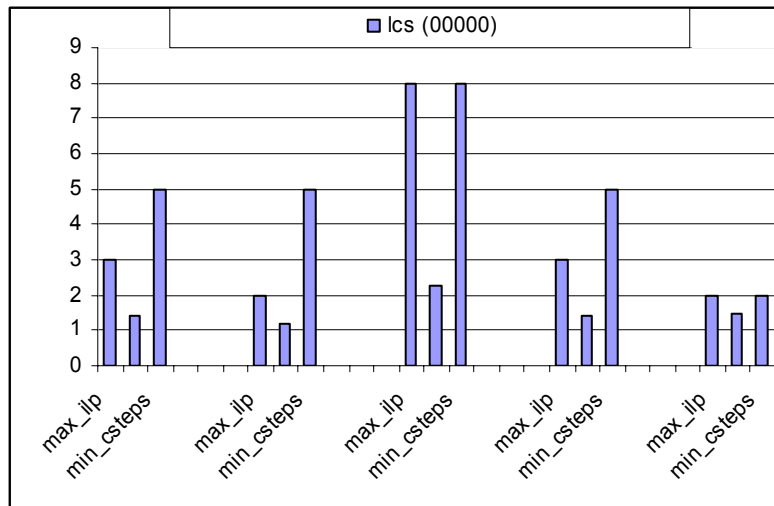
(c)



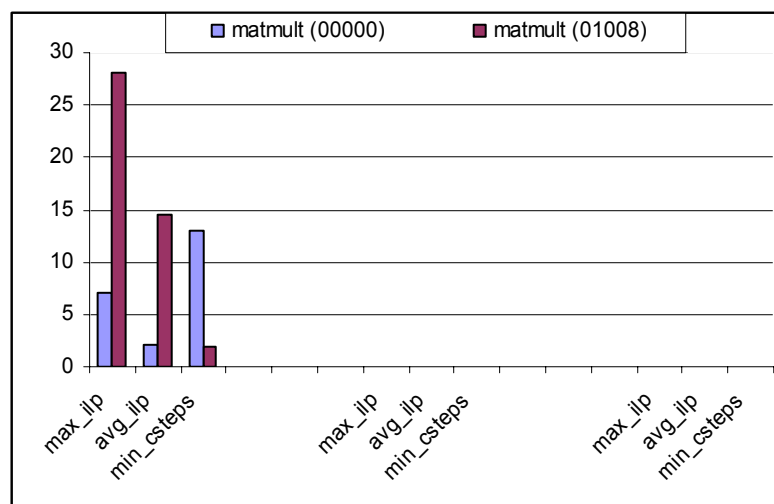
(d)



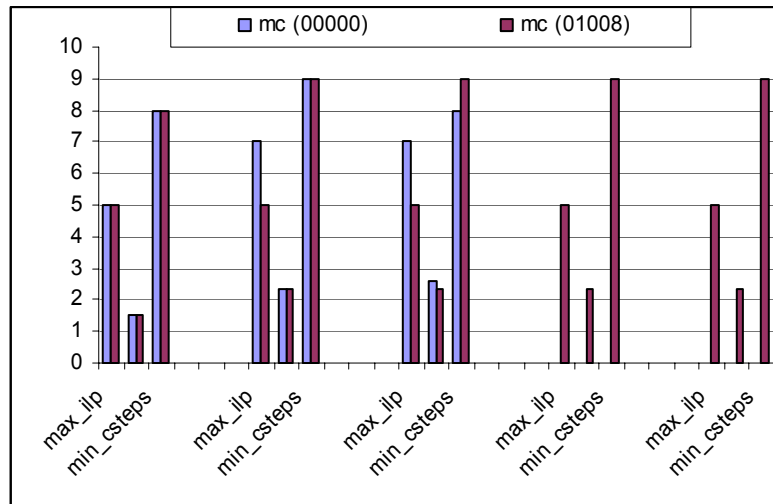
(e)



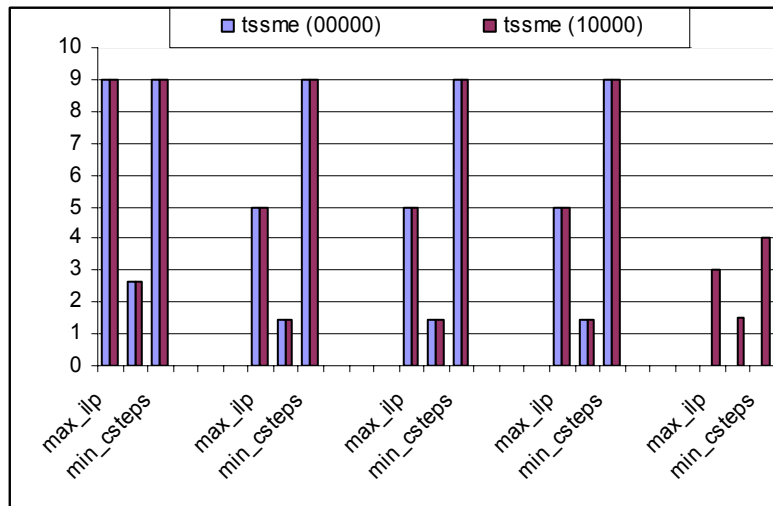
(f)



(g)



(h)



(i)

Σχήμα 4.5: Χαρακτηριστικά χρονοδρομολόγησης των «κρίσιμων» (hot) βασικών μπλοκ των εφαρμογών δοκιμής (BB 1 ως και 5). Απεικονίζεται η μέγιστη παραλληλία (max_ilp), μέση παραλληλία (avg_ilp), και ελάχιστος αριθμός βημάτων ελέγχου (min_csteps) για κάθε BB. (a) crc. (b) edgedet. (c) fsme. (d) fsp6c1. (e) kmppskip. (f) lcs. (g) matmult. (h) mc. (i) tssme.

Από τα αποτελέσματα του Σχήματος 4.5 διαπιστώνουμε τη θετική επίδραση του loop unrolling (σε μικρότερο βαθμό αυτό ισχύει και για τις βελτιστοποιήσεις if-conversion και inlining) στην αύξηση της διαθέσιμης εγγενούς παραλληλίας σε επίπεδο εντολών/μικρολειτουργιών στις εφαρμογές (fsp6c1, matmult). Για να είναι μεγαλύτερη η επίδραση του loop unrolling σε ορισμένες εφαρμογές (π.χ. fsme, tssme) θα έπρεπε να απομακρυνθούν αποφάσεις ελέγχου (εντολές cbr, ubr) από τα εσωτερικά των αντίστοιχων βρόχων. Κάτι τέτοιο είναι πάντα εφικτό μόνο για αρχιτεκτονικές πλήρως δεδηλωμένες (fully predicated) ενώ η αρχιτεκτονική SUiFvm είναι μερικώς δεδηλωμένη (partially predicated). Οι τρεις αυτές βελτιστοποιήσεις είναι τα κύρια «όπλα» του μεταγλωττιστή για την ανάδειξη μεγαλύτερης εγγενούς παραλληλίας των εφαρμογών.

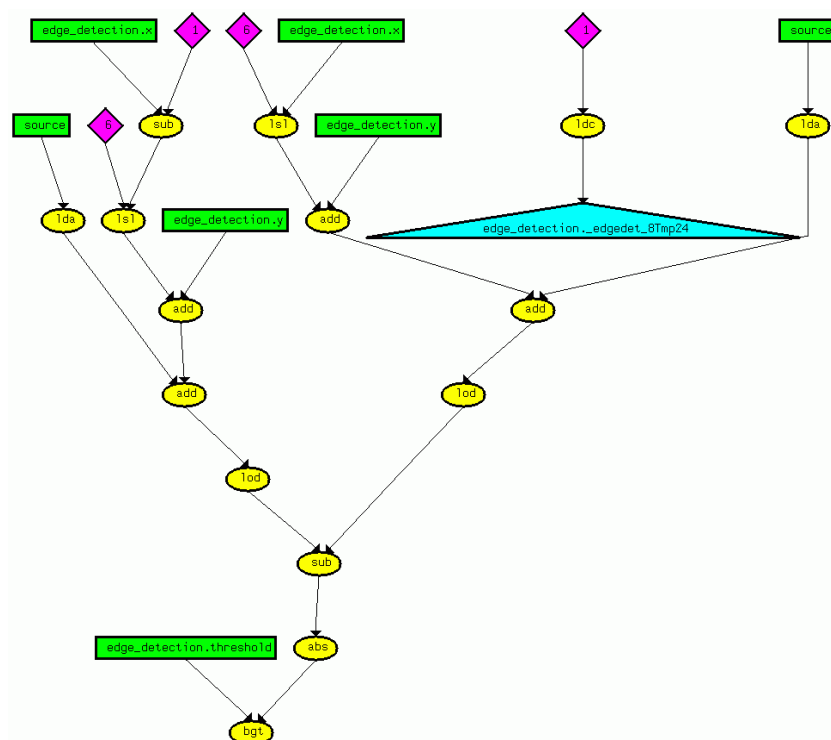
Το μέγιστο ILP (max_ilp) αποδίδει το ανώτερο όριο παραλληλίας (σε παράλληλες μικρολειτουργίες) που θα ήταν ιδανικά εκμεταλλεύσιμη για την εκτέλεση της εφαρμογής. Αυτή η πλήρης εκμετάλλευση είναι δυνατή μόνο σε ορισμένες περιοχές κώδικα (εντός των συγκεκριμένων και ίσως πουθενά αλλού στο ίδιο πρόγραμμα) και υπό προϋποθέσεις σχετικά

με τη διαθεσιμότητα λειτουργικών μονάδων και καταχωρητών. Το μέσο ILP (avg_ilp) δίνει καλύτερα ένα ποιοτικό μέτρο της πραγματικής παραλληλίας που μπορεί να επιτευχθεί σε όλη την έκταση του BB. Ένα γενικό συμπέρασμα είναι ότι σε όλο το φάσμα των εφαρμογών η μέγιστη χρήσιμη παραλληλία στο υλικό θα ήταν 4 χωρίς loop unrolling και 16 με loop unrolling κλιμακώνοντας στην εγγύτερη μεγαλύτερη δύναμη του 2.

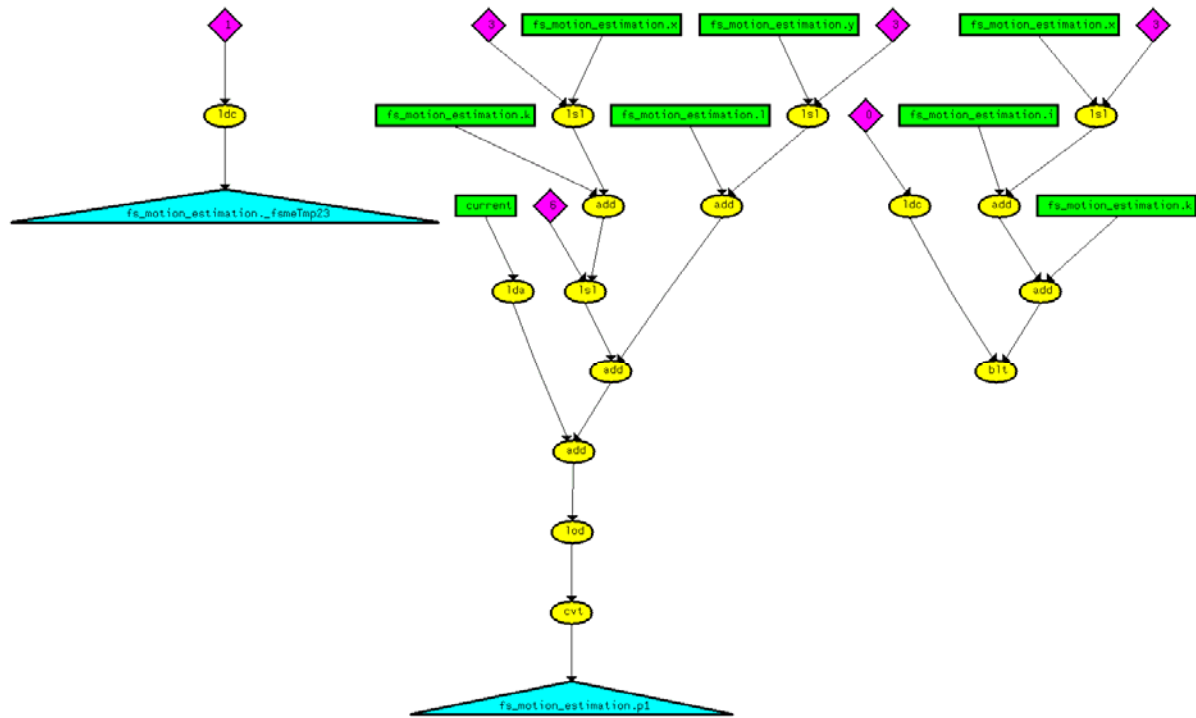
4.3.5. Οπτικοποίηση των γράφων εξάρτησης δεδομένων (DDG) για κρίσιμα βασικά μπλοκ των εφαρμογών

Κατά τη Φάση 2 αναπτύχθηκαν και εργαλεία που δεν έχουν να κάνουν με την εξαγωγή και ανάλυση της πληροφορίας που μπορεί να αναδειχθεί από τις εφαρμογές, αλλά με τη «διαχείριση» της. Στην κατηγορία αυτή ανήκουν και εργαλεία για τη δημιουργία διαφορετικών όψεων (facets) της πληροφορίας με την μετατροπή σε κατάλληλες διαμορφώσεις ώστε την εκμετάλλευση τρίτων εργαλείων και την ένταξή τους στη γενική ροή ανάλυσης εφαρμογών. Στο πλαίσιο αυτό δόθηκε ιδιαίτερη βαρύτητα στην πληροφορία των (τυποποιημένων) DDG για τα βασικά μπλοκ των εφαρμογών και για το λόγο αυτό συντάχθηκαν μετατροπείς (converters) **iseqinfo→vcg** [VCG] και **iseqinfo→dot** [GraphViz] που επιτρέπουν τη χρήση διαθέσιμων εργαλείων για τη διάταξη και οπτικοποίηση γράφων (graph layout and visualization). Οι διαμορφώσεις vcg και dot είναι ευρέως διαδεδομένες στο χώρο της ανάπτυξης μεταγλωττιστών, αναλυτών και γεννήτορων προγραμμάτων, εκσφαλμάτωσης κώδικα χωρίς οι χρήσεις τους να περιορίζεται σε αυτά.

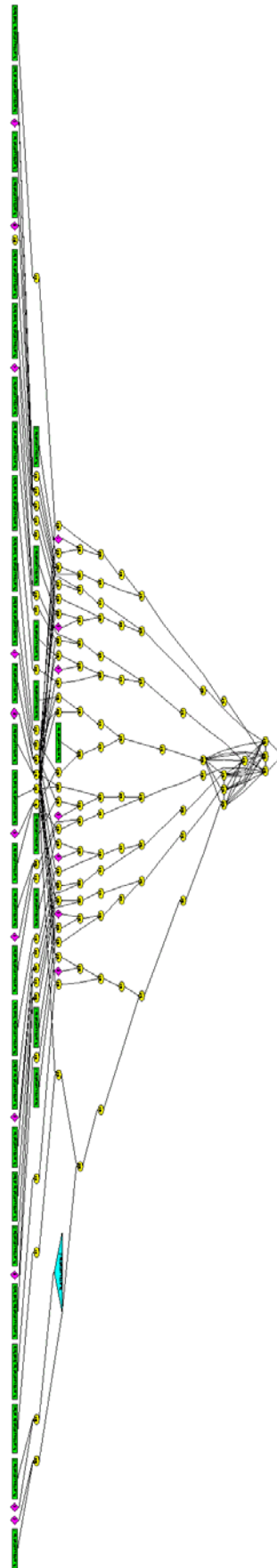
Στο Σχήμα 4.6 απεικονίζονται τα DDG για επιλεγμένα βασικά μπλοκ ορισμένων εφαρμογών. Για την διάταξη και οπτικοποίηση χρησιμοποιείται το λογισμικό VCG. Για την απεικόνιση χρησιμοποιούνται διαφορετικοί χρωματισμοί και σχήματα για εντολές, ορίσματα εισόδου/εξόδου και σταθερές.



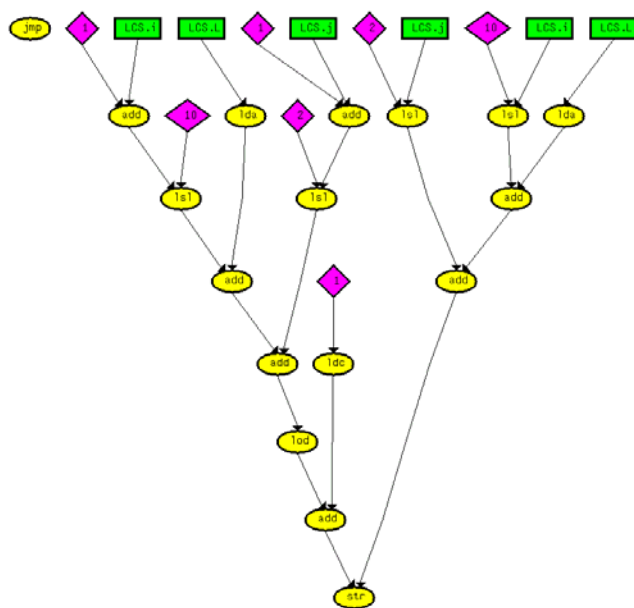
(a) edgedet-1-5



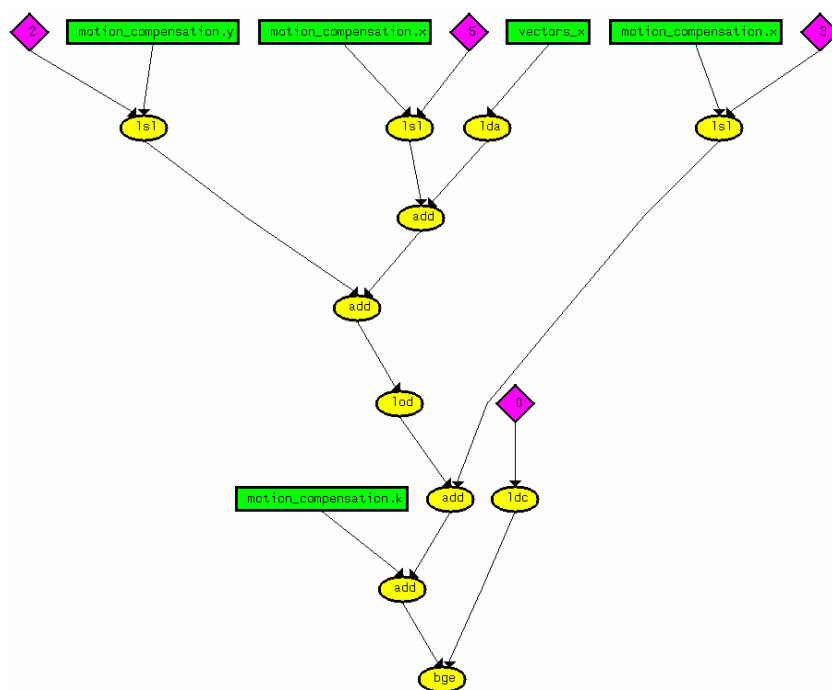
(b) fsme-1-13



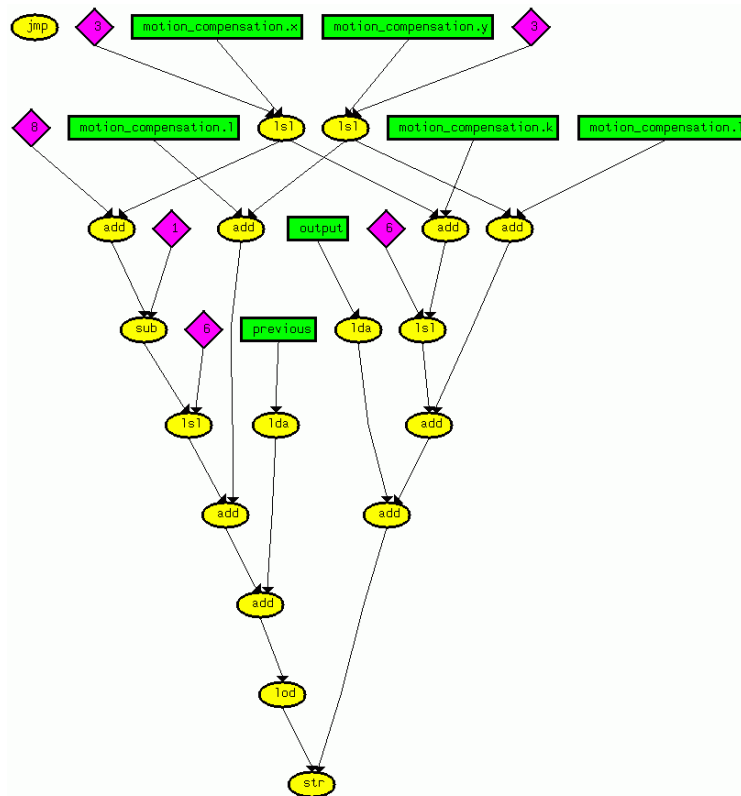
(c) fsp6c1-1-129



(d) lcs-1-11



(e) mc-1-9



(f) mc-1-14

Σχήμα 4.5: Οπτικοποιημένα DDG για επιλεγμένα κρίσιμα βασικά μπλοκ από τις εφαρμογές δοκιμής. (a) edgedet-1-5. (b) fsme-1-13. (c) fsp6c1-1-129. (d) lcs-1-11. (e) mc-1-9. (f) mc-1-14. Η διάταξη των γράφων έγινε με το πρόγραμμα VCG. Η χρωματική/σχηματική αποτύπωση γίνεται με τις εξής επιλογές: εντολές (κίτρινο/έλλειψη), ορίσματα εισόδου (πράσινο/παραλληλόγραμμο), σταθερές (πορφυρό/ρόμβος), ορίσματα εξόδου (κυανό/τρίγωνο).

5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ

Addressing mode	τρόπος διευθυνσιοδότησης
Application analysis	ανάλυση εφαρμογών
Argument	όρισμα (κατηγορήμα) συνάρτησης
Argument stack	στοίβα ορισμάτων
Assembler	συμβολομεταφραστής
Assembly	γλώσσα συμβολομεταφραστή
Attribute grammar	γραμματική ιδιοτήτων
Benchmarking	δοκιμασία επιδόσεων
Bitwidth analysis	ανάλυση εύρους ψηφίου
Branch prediction	πρόβλεψη διακλάδωσης
Cache analysis	ανάλυση κρυφής μνήμης
Code instrumentation	ενοργάνωση κώδικα
Code selection	επιλογή κώδικα
Compiler	μεταγλωττιστής
Compilation sequence	αλληλουχία μεταγλώττισης
Compilation space	πεδίο των μεταγλωττίσεων
Compiler infrastructure	υποδομή μεταγλωττιστή
Constant folding and propagation	δίπλωση και διάδοση σταθεράς
Control step	βήμα (λογικής) ελέγχου
Cycle accurate	ακρίβειας κύκλου



Cycle estimation	εκτίμηση κύκλου	
Debugging	εκσφαλμάτωση	
Disassembler	αποσυμβολομεταφραστής	
Edge	ακμή	
Exception	εξάιρεση	
Facet	όψη	
File interface	διεπαφή αρχείου	
Fully/partially predicated	πλήρως/μερικώς δεδηλωμένος	
Function inlining	διαγράμμιση συνάρτησης	
Graph	γράφος	
Graph layout and visualization	διάταξη και οπτικοποίηση γράφου	
Graph matching	ταίριασμα γράφων	
Hardware	υλικό	
Host machine	μηχάνημα ξενιστής	
If-conversion	μετατροπή δήλωσης «αν»	
Immediate	άμεσο όρισμα	
Interrupt	διακοπή	
Instruction accurate	ακρίβειας εντολής	
Instruction scheduler	χρονοδρομολογητής εντολών	
Intrinsic ILP	εγγενές ILP	
Line buffering and substitution	απομόνωση και αντικατάσταση γραμμής	
Linker	συνδέτης	
Loop unrolling	ξετύλιγμα βρόχων	
Multimedia	πολυμέσα	
Multiplexer	πολυπλέκτης	
Node	κόμβος	
Object code	αντικείμενος κώδικας	
Opcode	κώδικας λειτουργίας	
Pass	εδώ: πέρασμα μεταγλωττιστή	
Pattern matching	ταίριασμα μοτίβου	
Peephole optimization	βελτιστοποίηση «κλειδαρότρυπας»	
Pipeline stage	βαθμίδα διοχέτευσης	
Pointer	δείκτης (μεταβλητή δεικτοδότησης)	
Predicated execution	δεδηλωμένη εκτέλεση	
Ranking	κατάταξη	
Rapid retargeting	ταχεία επαναστόχευση	
Register allocator	κατανεμητής καταχωρητών	
Retargetable compiler	επαναστοχευόμενος μεταγλωττιστής	
Scheduling	(χρονο-) δρομολόγηση	
Signed	προσημασμένος	
Simulator	προσομοιωτής	
Software	λογισμικό	
Software development toolchain	αλυσίδα εργαλείων ανάπτυξης λογισμικού	
Software pipelining	λογισμική διοχέτευση	
Superscalar	υπερβαθμωτός	
(non) terminal	(μη) τερματικό(ς)	
Textual form	μορφή κειμένου	
Topological sort	τοπολογική ταξινόμηση	
Tree pattern matching	ταίριασμα μοτίβου δένδρου	
Tree traversal	διάβαση δένδρου	
Unsigned	απρόσημος	
Variadic	μεταβλητού αριθμού ορισμάτων (κατηγορημάτων)	
Vectorization	διανυσματοποίηση	
Virtual register	εικονικός καταχωρητής	
ADL	Architecture Description Language	Γλώσσα Περιγραφής Αρχιτεκτονικής
API	Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογής
ASAP	As Soon As Possible	
ASIP	Application-Specific Instruction-set Processor	Επεξεργαστής Συνόλου Εντολών Ειδικού Σκοπού

AST	Abstract Syntax Tree	Αφηρημένο Συντακτικό Δένδρο
CCDG	Composed Control Dependence Graph	Συντεθειμένος Γράφος Εξάρτησης Ελέγχου
CDFG	Control-Data Flow Graph	Γράφος Ροής Ελέγχου-Δεδομένων
CFG	Control Flow Graph	Γράφος Ροής Ελέγχου
CISC	Complex Instruction Set Computer	Υπολογιστής Σύνθετου Συνόλου Εντολών
CSE	Common Subexpression Elimination	Εξουδετέρωση Κοινής Υποεκφράσεως
DAG	Directed Acyclic Graph	Κατευθυνόμενος Ακυκλικός Γράφος
DDG	Data Dependence Graph	Γράφος Εξάρτησης Δεδομένων
DCE	Dead Code Elimination	Εξουδετέρωση Νεκρού Κώδικα
DFG	Data Flow Graph	Γράφος Ροής Δεδομένων
DFT	Data Flow Tree	Δένδρο Ροής Δεδομένων
DSP	Digital Signal Processor/Processing	Επεξεργαστής/Επεξεργασία Ψηφιακού Σήματος
(E)BNF	(Extended) Backus-Naur Form	(Εκτεταμένη) Μορφή Backus-Naur
FP	Frame Pointer	Δείκτης Πλαισίου
GCC	GNU Compiler-Compiler	
HDL	Hardware Description Language	Γλώσσα Περιγραφής Υλικού
HLL	High Level Language	Γλώσσα Υψηλού Επιπέδου
HLS	High-Level Synthesis	Σύνθεση Υψηλού Επιπέδου
ILP	Instruction-Level Parallelism	Παραλληλία Επιπέδου Εντολής
IR	Intermediate Representation	Ενδιάμεση Αναπαράσταση (μεταγλωττιστή)
MMU	Memory Management Unit	Μονάδα Διαχείρισης Μνήμης
PDG	Program Dependence Graph	Γράφος Εξάρτησης Προγράμματος
RA	Return Address	Διεύθυνση Επιστροφής
RISC	Reduced Instruction Set Computer	Υπολογιστής Μειωμένου Συνόλου Εντολών
RISP	Reconfigurable Instruction Set Processor	Επαναπροσδιορίσιμος Επεξεργαστής Συνόλου Εντολών
SP	Stack Pointer	Δείκτης Στοίβας
SSA	Static Single Assignment	Στατική Απλή Ανάθεση
SSI	Static Single Information	Στατική Απλή Πληροφορία
SUIFrm	SUIF real machine	Πραγματική Μηχανή SUIF
SUIFvm	SUIF virtual machine	Εικονική Μηχανή SUIF
TAC	Three Address Code	Κώδικας Τριών Διευθύνσεων
VLIW	Very Long Instruction Word	Πολύ Μεγάλο Μήκος Εντολής
WPP	Whole Program Optimization	Βελτιστοποίηση Ολόκληρου Προγράμματος
ΛΣ	Operating System	Λειτουργικό Σύστημα
ΤΔ	Data Type	Τύπος Δεδομένων

5. ΑΝΑΦΟΡΕΣ

[Aho86] A. V. Aho, R. Sethi and J. D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, Reading, MA, USA, 1986.

[ArchC] The ArchC resource center. <http://www.archc.org>

[Aus02] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," IEEE Computer, Vol. 35, No. 2, pp. 59-67, Feb. 2002.

[Aze05] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araujo, C. Araujo, and E. Barros, "The ArchC architecture description language and tools," International Journal of Parallel Programming, Vol. 33, No. 5, pp. 453-484, Oct. 2005.

[Binutils] GNU Binutils. <http://sources.redhat.com/binutils/>

[bison] Bison homepage. <http://www.gnu.org/software/bison/bison.html>

[Bou01] D. Boulytchev, and D. Lomov, "An empirical study of retargetable compilers," In proceedings of the 4th International Conference on Perspective of System Informatics, Akademgorodok, Novosibirsk, Russia, July 2001.

[Bur97] D.C. Burger and T.M. Austin, "The SimpleScalar tool set, version 2.0", Technical report CS-TR-1997-1342, Computer Sciences Department, University of Wisconsin-Madison, Jun. 1997.

- [CDFGtool] CDFG toolset. <http://poppy.snu.ac.kr/CDFG/cdfg.html>
- [CDFGtool-guide] J. Jeon, Y. Ahn, and K. Choi, "CDFG Toolkit User's Guide", Technical Report N. SNU-EE-TR-2002-8, School of Electronic Engineering, Seoul National University, August 2002.
- [ChaLec] C. Charras, and T. Lecroq, Handbook of Exact String-Matching, draft version.
- [Cla03] N. Clark, H. Zhong, W. Tang, and S. Mahlke, "Automatic design of application specific instruction set extensions through dataflow graph exploration," International Journal of Parallel Programming, Vol. 31, No. 6, pp. 429-449, Dec. 2003.
- [copt] C.F. Fraser. Copt peephole optimizer. <http://www.well.com/~cwf/pro/vita.htm>
- [Coware] CoWare, Inc.. <http://www.coware.com>
- [Cyt91] R. Cytron, J. Ferrante, B. Rosen, M. Wegman, and K. Zadeck, "Efficiently computing static single assignment form and the control dependence graph," ACM Transactions on Programming Languages and Systems, Vol. 13, No. 4, pp. 451-490, October 1991.
- [EBNF] ISO/IEC 14977 international standard. Extended BNF syntactic metalanguage. Dec. 1996.
- [EPFL-ARM] G. Theoduloz and D. S. Garcia, Machine-SUIF backend for the ARM architecture. http://lap.epfl.ch/dev/machsuiif/arm_backend/
- [EPFL-passes] Optimization passes for Machine-SUIF. http://lap.epfl.ch/dev/machsuiif/opt_passes/
- [Ertl99] M. A. Ertl, "Optimal code selection in DAGs," ACM Symposium on Principles of Programming Languages (POPL'99).
- [EXPRESS] Express Group SUIF passes. <http://express.ece.ucsb.edu/software.html>
- [Fer87] J. Ferrante, K. J. Ottenstein and J. D. Warren, "The program dependence graph and its use in optimization," ACM Transactions on Programming Languages and Systems, Vol. 9, No. 3, pp. 319-349, July 1987.
- [flex] Flex (The Fast Lexical Analyzer) homepage. <http://flex.sourceforge.net>
- [Fra95] C. Fraser and D. Hanson, A Retargetable C compiler: Design and implementation, Addison-Wesley, 1995.
- [GCC] The GNU Compiler Collection homepage. <http://gcc.gnu.org>
- [Graphviz] Graphviz: Graph visualization software. <http://www.graphviz.org>
- [Gri03] M. Gries, "Methods for Evaluating and Covering the Design Space during Early Design Development," Technical Memorandum, UCB/ERL M03/32, CAD-Group, Electronics Research Laboratory, University of California at Berkeley, CA, USA, Aug. 2003.
- [Hen96] J.L. Hennessy, and D.A. Patterson, Computer Architecture: a Quantitative Approach, 2nd edition, Morgan Kaufmann Publishers, San Fransisco, CA, 1996.
- [Hof02] A. Hoffmann, H. Meyr, and R. Leupers, Architecture exploration for embedded processors with LISA, Kluwer Academic Publishers, Dec. 2002.
- [IBURG] IBURG, a tree parser generator. <http://www.cs.princeton.edu/software/iburg/>
- [Kav04] N. Kavvadias and S. Nikolaidis, "Application Analysis with Integrated Identification of Complex Instructions for Configurable Processors," Proc. of the 14th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation, pp. 633-642, September 15-17, 2004, Santorini, Greece.
- [Kog81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing", Proc. of the National Telecommunications Conference, pp. G5.3.1-G5.3.5, New Orleans, LA, USA, Nov. 1981.
- [LANCE] LANCE C compiler. <http://www.lancecompiler.com>
- [LCC] LCC retargetable C compiler. <http://www.cs.princeton.edu/software/lcc/>
- [Leu03] R. Leupers, O. Whalen, M. Hohenauer, T. Kogel, and P. Marwedel, "An executable intermediate representation for retargetable compilation and high-level code optimization," Proceedings of the 1st International Workshop on Systems, Architectures, Modeling, and Simulation, pp. 120-125, July 2003, Samos, Greece.
- [MachSUIF] Machine-SUIF research compiler. <http://www.eecs.harvard.edu/hube/research/machsuiif.html>
- [MIPS] MIPS Technologies Inc.. <http://www.mips.com>
- [MonoBURG] MonoBURG homepage. <http://www.lrde.epita.fr/cgi-bin/twiki/view/Tiger/MonoBURG>
- [Nios-II] Altera Nios-II home page. <http://www.altera.com/products/ip/processors/nios2/>
- [OpenIMPACT] OpenIMPACT at UIUC, an open-source Itanium compiler. <http://www.gelato.uiuc.edu/>

- [Pel88] E. Pelegri-Llopart, and S. L. Graham, "Optimal code generation for expression trees: An application of BURS theory," In proceedings of the 15th Annual ACM Symposium on Principles of Programming Languages, pp. 294-308, San Diego, California, USA, Jan. 1988.
- [Proebstring] T. A. Proebsting: Proebsting's Law: Compiler Advances Double Computing Power Every 18 Years. <http://www.research.microsoft.com/~toddp/papers/law.htm>
- [Rol03] L. Rolaz, "An implementation of if-conversion using select instructions for Machine SUIF", if-conversion pass user manual, Swiss Federal Institute of Technology (EPFL), Processor Architecture Laboratory, Lausanne, Switzerland, Mar. 2003.
- [San94] G. Sander, "Graph layout through the VCG tool," in Proceedings DIMACS International Workshop on Graph Drawing, pp. 194-205, 1994, Berlin, Germany.
- [She04] T. P. Shevlin, "Composed Control Dependence Graph Generator," M.Sc. thesis, Department of Computer Science, Tufts University, Medford, Massachusetts, USA, Sep. 2004.
- [SimpleScalar] SimpleScalar simulation framework homepage. <http://www.simplescalar.com>
- [Sin04] J. Singer, Static Single Information form in Machine SUIF, machsuifssi user manual, version 0.1, Mar. 2004.
- [Smi02] M.D. Smith and G. Holloway, "An introduction to Machine SUIF and its portable libraries for analysis and optimization," Technical report, 2.02.07.15 edition, Division of Engineering and Applied Sciences, Harvard University, USA, 2002.
- [SPAM] The SPAM project. (now defunct). <http://www.ee.princeton.edu/spam/>
- [SUIF] SUIF compiler infrastructure. <http://suif.stanford.edu/suif/suif2/>
- [SystemC] SystemC community homepage. <http://www.systemc.org>
- [The05] G. Theoduloz, "Optimization of a Machine SUIF compiler backend for the ARM architecture", Technical Report, Swiss Federal Institute of Technology (EPFL), Processor Architecture Laboratory, Lausanne, Switzerland, Winter 2004-2005.
- [Trimaran] Trimaran Consortium, The Trimaran Compiler Research Infrastructure for Instruction Level Parallelism. <http://www.trimaran.org>
- [VCG] VCG website. <http://rw4.cs.ui-sb.de/~sander/html/gsvcg1.html>
- [VPO] VPO (Very Portable Optimizer) homepage: <http://www.cs.virginia.edu/zephyr/vpo/>