

Μεταγλωττιστές II

Ανασκόπηση του μαθήματος και θέματα πρακτικής
εξάσκησης

Νικόλαος Καββαδίας
nkavn@uop.gr

26 Ιανουαρίου 2011

Σκιαγράφηση της διάλεξης

- Παραλειπόμενα
- Αναδρομή στο περιεχόμενο του μαθήματος
- Ενδεικτικά θέματα εξετάσεων (θεωρία και ασκήσεις)
- Θέματα για πρακτική εξάσκηση (ασκήσεις)
- Άλλα θέματα (κρίσεως και σύνθεσης)

Η έννοια της μεταγλώττισης και η δομή του μεταγλωττιστή

- Μετάφραση από μία πηγαία γλώσσα η οποία διέπεται από γραμματική σε κάποια τελική γλώσσα στο ίδιο ή διαφορετικό επίπεδο αφαιρέσης
- **compiler**: το λογισμικό που επιτελεί τη μετάφραση προγραμμάτων σε γλώσσα προγραμματισμού υψηλού επιπέδου (HLL) στο επίπεδο του κώδικα μιας πραγματικής ή εικονικής μηχανής
- Η διαδικασία της μεταγλώττισης μπορεί να χωριστεί στη φάση της ‘ανάλυσης’ και στη φάση της ‘σύνθεσης’
 - **ΑΝΑΛΥΣΗ**: αποδόμηση και κατανόηση του πηγαίου προγράμματος
 - **ΣΥΝΘΕΣΗ**: κατασκευή του αποτελέσματος στην τελική γλώσσα διατηρώντας σημασιολογική ισοδυναμία με το πηγαίο πρόγραμμα
- Οι πρακτικοί μεταγλωττιστές αποτελούνται από πολλά διαδοχικά τμήματα (περάσματα)

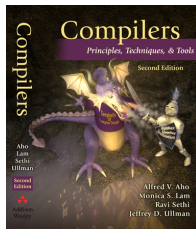
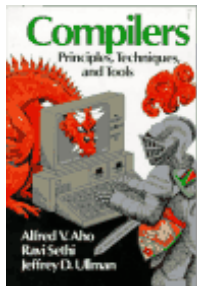
Γενική ορολογία από την ανάπτυξη μεταγλωττιστών (Δ. Σπινέλλης)

- **Κειμενογράφος/Διορθωτής (Editor):**
Επιτρέπει τη συγγραφή και την αλλαγή του προγράμματος
- **Προεπεξεργαστής (Preprocessor):**
Επεξεργάζεται το πρόγραμμα εκτελώντας απλούς συμβολικούς μετασχηματισμούς και παράγει ένα ισοδύναμο πρόγραμμα (αφορά τις C, C++, Fortran)
- **Συμβολομεταφραστής (Assembler):**
Μετατρέπει τη συμβολική γλώσσα του επεξεργαστή σε γλώσσα μηχανής
- **Μεταγλωττιστής (Compiler):**
Μεταφράζει μια γλώσσα υψηλού επιπέδου σε γλώσσα επιπέδου μηχανής
- **Διερμηνευτής (Interpreter):**
Εκτελεί άμεσα ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου
- **Συνδέτης (Linker):**
Συρράφει τμήματα ενός προγράμματος που έχουν μεταγλωττιστεί ξεχωριστά σε ένα ενιαίο πρόγραμμα
- **Φορτωτής (Loader):**
Φορτώνει το πρόγραμμα στη μνήμη του επεξεργαστή διορθώνοντας αναφορές σε θέσεις μνήμης εντολών και δεδομένων του προγράμματος
- **Αποσφαλματωτής (Debugger):**
Επιτρέπει την εκτέλεση του προγράμματος βήμα-βήμα με σκοπό την ανίχνευση λαθών που μπορεί να περιέχει το πρόγραμμα

Η εργαλειοθήκη του σχεδιαστή μεταγλωττιστών

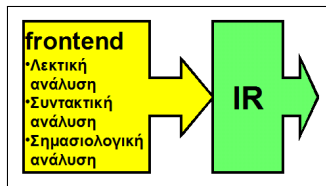
- Κεμενογράφος/Διορθωτής:
vi, emacs, Geany, Context, Prism Editor, Notepad-++
- Λεκτική/συντακτική ανάλυση:
lex+yacc, flex+bison, ANTLR (πρώην PCCTS), GOLD Parser Builder
- Συμβολομεταφραστής-Συνδέτης-Αποσυμβολομεταφραστής:
binutils (as, ld, objdump)
- Μεταγλωττιστής (Compiler):
GCC, LCC, LLVM, COINS, Phoenix, PCC, Trimaran, SUIF/Machine-SUIF
- Πρότυπη βιβλιοθήκη της C:
glibc, newlib, uclibc, dietlibc
- Αποσφαλματωτής (Debugger):
GDB
- Γεννίτορες γεννητόρων κώδικα:
BURG, IBURG, LBURG, OLIVE
- Οπτικοποίηση γραφών:
Graphviz, VCG
- Άλλα εργαλεία:
sparse, Aha!, superopt, copt

Χρήσιμα βιβλία



Τύποι IR

- Η εξαγωγή της IR (ενδιάμεση αναπαράσταση) είναι το αποτέλεσμα της λεκτικής, συντακτικής και σημασιολογικής ανάλυσης του πηγαίου προγράμματος



- Επίπεδη διαμόρφωση σε μορφή εντολών: κώδικας τριών διευθύνσεων (Three-Address Code, συχνά 3AC ή TAC)
 - Απλή δομή, κατάλληλη για βελτιστοποιήσεις
- Διαμόρφωση τύπου γράφου: Γράφος Ροής Ελέγχου-Δεδομένων (CDFG: Control-Data Flow Graph)
 - Περισσότερο αποκαλυπτική για τα χαρακτηριστικά του πηγαίου προγράμματος, κατάλληλη για γέννηση κώδικα

Σύνοψη του μαθήματος

- 1 Η οργάνωση του δομημένου μεταγλωττιστή
- 2 Γέννηση ενδιάμεσης αναπαράστασης
- 3 Επιλογή κώδικα
- 4 Καταμερισμός καταχωρητών
- 5 Βελτιστοποιήσεις ανεξάρτητες από την αρχιτεκτονική
- 6 Χρονοπρογραμματισμός κώδικα και βελτιστοποιήσεις εξαρτημένες από την αρχιτεκτονική
- 7 Βελτιστοποιήσεις για την εκμετάλλευση της παραλληλίας και ανάδειξη της τοπικότητας
- 8 Γέννηση τελικού κώδικα
- 9 Επαναστοχεύσιμοι μεταγλωττιστές

Δ1: Οργάνωση του μεταγλωττιστή

- Στοιχεία από το διαδικαστικό προγραμματισμό (διαγράμματα ροής, ANSI C)
- Παρουσίαση της οργάνωσης του δομημένου μεταγλωττιστή
- Η χρησιμότητα της ενδιάμεσης αναπαράστασης
- Πρακτικοί μεταγλωττιστές
- Ενδεικτικά θέματα
 - 1) Να δοθεί το σχηματικό διάγραμμα του τυπικού σχεδιασμού ενός μεταγλωττιστή, να ονομαστεί κάθε επιμέρους τμήμα του και να δοθεί σύντομη περιγραφή της λειτουργίας του.
 - 2) Ποια η λειτουργία του πίνακα συμβόλων (σύντομα);
 - 3) Ποια τα πλεονεκτήματα της χρήσης ενδιάμεσης αναπαράστασης στο σχεδιασμό ενός επαναστοχεύσιμου μεταγλωττιστή; Να δοθεί αριθμητικό παράδειγμα για την περίπτωση μεταγλωττιστή ο οποίος δέχεται τις πηγαίες γλώσσες ANSI C, C++, και Pascal και παράγει κώδικα στις γλώσσες συμβολομεταφραστή για τις αρχιτεκτονικές x86, MIPS, ARM και PowerPC.

Δ2: Γέννηση ενδιάμεσης αναπαράστασης

- Αποδόμηση σύνθετων εκφράσεων της ANSI C
- Βασικό μπλοκ - αναπαράσταση DAG
- Κώδικας τριών διευθύνσεων (three address code)
- Μεταγλώττιση πηγαίου κώδικα σε TAC
- Γράφοι εξάρτησης δεδομένων
- CFG και CDFG
- Ορισμός της SSA
- Κατασκευή SSA

Δ2: Ενδεικτικά θέματα

- 1) Τι είναι η μορφή Στατικής Απλής Ανάθεσης (SSA) και ποια η κύρια ιδιότητά της;
- 2) Τι είναι ο γράφος ροής ελέγχου (CFG) και τι αναπαριστά; Τι είναι βασικό μπλοκ σε ένα γράφο ροής ελέγχου και ποια τα χαρακτηριστικά του; Δώστε ένα παράδειγμα βασικού μπλοκ (μέχρι 7 εντολές) με κώδικα τριών διευθύνσεων (TAC).
- 3) Ο παρακάτω ANSI C κώδικας περιγράφει έναν αλγόριθμο υπολογισμού του παραγοντικού ($n!$) του μη-αρνητικού ακέραιου αριθμού n . Να παραχθεί ο γράφος ροής ελέγχου-δεδομένων (CDFG) για τον αλγόριθμο.

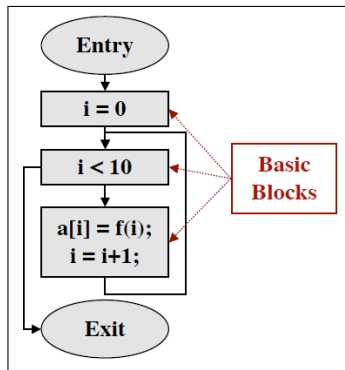
```
if (n == 0) {  
    res = 1;  
}  
else {  
    res = 1;  
    for (i = 1; i <= n; i++) {  
        res = res * i;  
    }  
}
```

Δ2: CFG από δομημένο πηγαίο κώδικα

■ Κώδικας ANSI C

```
extern int f(int);  
  
int main(void)  
{  
    int i;  
    int *a;  
  
    for (i = 0; i < 10; i++)  
    {  
        a[i] = f(i);  
    }  
}
```

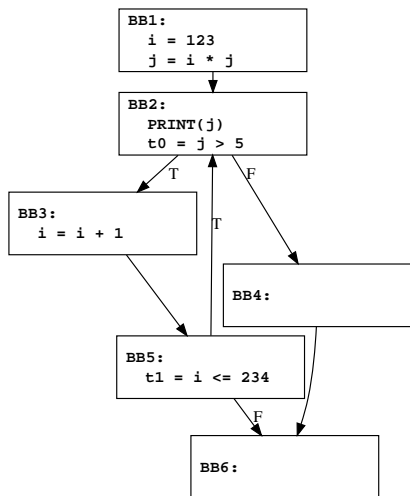
■ Το CFG της συνάρτησης main με δηλώσεις C



Δ2: Παράδειγμα 1 στην κατασκευή SSA (από την εργασία των Ayccock-Horspool)

■ Πηγαίο πρόγραμμα

```
i = 123;  
j = i * j;  
do  
{  
  PRINT(j);  
  if (j > 5)  
  {  
    i = i + 1;  
  }  
  else  
  {  
    break;  
  }  
} while (i <= 234);
```



Αντιπαράθεση non-SSA και SSA IR ως TAC

■ Non-SSA IR

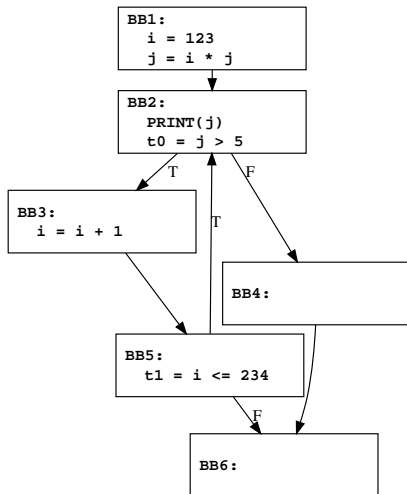
```
BB1:
  i = 123;
  j = i * j;
BB2:
  PRINT(j);
  t0 = j > 5;
  if (t0) goto BB3; else goto BB4;
BB3:
  i = i + 1;
  goto BB5;
BB4:
  goto BB6;
BB5:
  t1 = i <= 234;
  if (t1) goto BB2; else goto BB6;
BB6:
```

■ SSA IR

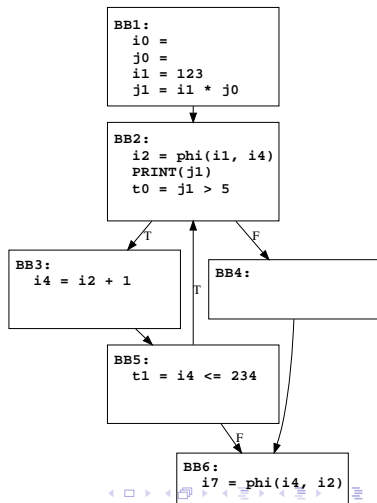
```
BB1:
  i1 = 123;
  j1 = i1 * j0;
BB2:
  i2 = phi(i1, i4);
  PRINT(j1);
  t0 = j1 > 5;
  if (t0) goto BB3; else goto BB4;
BB3:
  i4 = i2 + 1;
  goto BB5;
BB4:
  goto BB6;
BB5:
  t1 = i4 <= 234;
  if (t1) goto BB2; else goto BB5;
BB6:
  i7 = phi(i4, i2);
```

Αντιπαράθεση non-SSA και SSA IR ως CFG

■ Non-SSA CFG



■ SSA CFG



Προτεινόμενα θέματα στην κατασκευή SSA (για εξάσκηση)

■ Ακολουθία Fibonacci

```
BB0:
x = n;
f0 = 0;
f1 = 1;
res = f0;
if (x <= 0) {goto BB4;}
else {goto BB1;}
BB1:
res = f1;
if (x == 1) {goto BB4;}
else {goto BB2;}
BB2:
k = 2;
goto BB3;
BB3:
f = f1 + f0;
f0 = f1;
f1 = f;
res = f;
k = k + 1;
if (k <= x) {goto BB3;}
else {goto BB4;}
BB4:
```

■ Απαρίθμηση πληθυσμού

```
BB1:
data = inp;
count = 0;
goto BB2;
BB2:
temp = data & 1;
count = count + temp;
data = data >> 1;
if (data == 0) {goto BB3;}
else {goto BB2;}
BB3:
```

Δ3: Επιλογή κώδικα

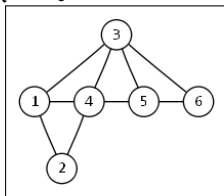
- Η έννοια της κοινής υποεκφράσεως
- Μη βέλτιστη πλακόστρωση δένδρου ροής δεδομένων για την επιλογή κώδικα
- Σχεδιασμός AST από κειμενική αναπαράσταση DFT
- Ενδεικτικά θέματα
 - 1) Να περιγραφεί η αρχή λειτουργίας της επιλογής κώδικα με κάλυψη δένδρου.
 - 2) Υπάρχουν τεχνικές οι οποίες επιτυγχάνουν βέλτιστη επίλυση του προβλήματος της κάλυψης δένδρου για την επιλογή κώδικα; Αν ναι, αναφέρετε μία τέτοια τεχνική και ένα λογισμικό εργαλείο το οποίο να την χρησιμοποιεί.

Δ4: Καταμερισμός καταχωρητών

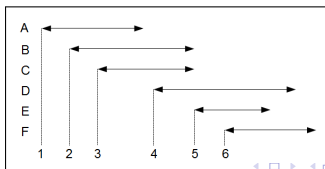
- Διαστήματα ζωής: εξαγωγή από κώδικα TAC
- Καθολικός καταμερισμός καταχωρητών
- Χρωματισμός γράφου - Γράφοι παρεμβολής
- Περιοχές ζωής - η διαφορά τους από τα διαστήματα ζωής
- Ο αλγόριθμος του Chaitin και πως εφαρμόζεται
- Ο αλγόριθμος γραμμικής σάρωσης: περιγραφή και εφαρμογή
- Ενδεικτικά θέματα (θεωρία)
 - 1) Τι γνωρίζετε για την περιοχή ζωής και για το διάστημα ζωής; Ποιες οι διαφορές τους;
 - 2) Αναλύστε την υπολογιστική πολυπλοκότητα του αλγορίθμου γραμμικής σάρωσης για τον καταμερισμό καταχωρητών.

Ενδεικτικά θέματα: Καταμερισμός καταχωρητών

- Να πραγματοποιηθεί καταμερισμός καταχωρητών:
 - α) Με τον αλγόριθμο χρωματισμού γράφου ($k = 3$) για το γράφο παρεμβολής του σχήματος.



- β) Με τον αλγόριθμο της γραμμικής σάρωσης για τους χρόνους ζωής (A-F). Ο αριθμός των διαθέσιμων φυσικών καταχωρητών είναι $R = 3$.



Παράδειγμα εξαγωγής διαστημάτων χρόνου ζωής μεταβλητών

- Βασικό μπλοκ του παραδείγματος

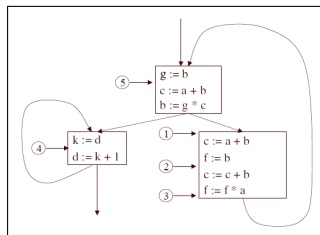
```
1 b = 1;  
2 c = 2;  
3 a = b + c;  
4 d = a * 2;  
5 e = b / 3;  
6 return (e - d);
```

- Υπολογισμός των διαστημάτων ζωής των μεταβλητών

	1	2	3	4	5	6
a			X	X		
b	X	X	X	X	X	
c		X	X			
d				X	X	X
e					X	X

Θέματα εξάσκησης: Ανάλυση χρόνου ζωής

- Έστω το παρακάτω CFG. Να δοθούν τα σύνολα ζωντανών μεταβλητών στα σημεία 1 ως 5.



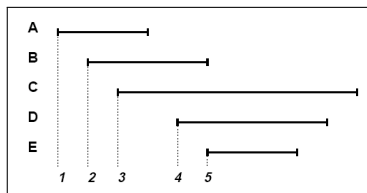
■ Απάντηση

Live at 1 = {a, b, d}
Live at 2 = {a, b, c, d, f}
Live at 3 = {a, b, d}
Live at 4 = {k}
Live at 5 = {a, b, d, g}

- i** Μεταβλητές που μόνο διαβάζονται είναι ζωντανές πριν το σημείο εισόδου
- i** Ελέγξτε όλες τις διαδρομές

Παράδειγμα εφαρμογής του αλγορίθμου γραμμικής σάρωσης για τον καταμερισμό καταχωρητών

- Έστω οι προσωρινές μεταβλητές A, B, C, D, E και τα αντίστοιχα διαστήματα 1 ως 5 του σχήματος και $R = 2$



Έναρξη I1	active = $\langle A \rangle$	$R0 \leftarrow A$
Έναρξη I2	active = $\langle A, B \rangle$	$R1 \leftarrow B$
Έναρξη I3	active = $\langle A, B \rangle$	$R0 \leftarrow A, R1 \leftarrow B, \text{spill } C$
Έναρξη I4	active = $\langle B, D \rangle$	$R0 \leftarrow D, R1 \leftarrow B, C$ spilled
Έναρξη I5	active = $\langle D, E \rangle$	$R0 \leftarrow D, R1 \leftarrow E, C$ spilled

Δ5: Βελτιστοποιήσεις ανεξάρτητες από την αρχιτεκτονική

- Ο βελτιστοποιητής στο πλαίσιο του δομημένου μεταγλωττιστή
- Βασικές διαφορές μεταξύ βελτιστοποιήσεων υψηλού και χαμηλού επιπέδου - Παραδείγματα
- Η εφαρμογή όλων των βαθμωτών βελτιστοποιήσεων
- Σχεδιασμός δένδρου κυριαρχίας (όχι βέλτιστος αλγόριθμος)
- Ενδεικτικά θέματα (θεωρία)
 - 1) Εφαρμοστέ διαδοχικά δίπλωση σταθεράς, διάδοση σταθεράς, αλγεβρικές απλοποιήσεις και εξουδετέρωση κοινής υποεκφράσεως στο παρακάτω τμήμα κώδικα.

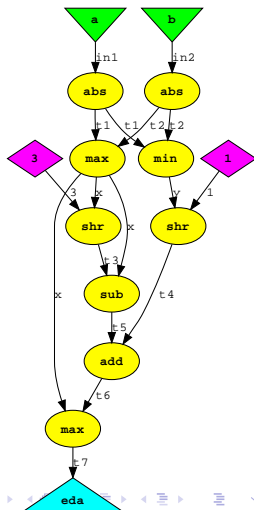
```
if (k == 0) {  
    a = 11 + 1;  
    b = a;  
    c = (b + e) * 1024;  
    d = e + b;  
} else {  
    x = 9 * a + c / 2;  
}
```

Δ6: Χρονοπρογραμματισμός κώδικα και βελτιστοποιήσεις εξαρτημένες από την αρχιτεκτονική

- Εξαρτήσεις εντολών
- Αρχές στατικού και δυναμικού χρονοπρογραμματισμού
- Ο αλγόριθμος ASAP
- Ο αλγόριθμος χρονοπρογραμματισμού λίστας
- Ενδεικτικά θέματα θεωρίας
 - 1) Να αναφέρετε τις αρχές που διέπουν και τα χαρακτηριστικά (ομοιότητες, διαφορές) του στατικού και του δυναμικού χρονοπρογραμματισμού.
 - 2) Τι είναι η εξάρτηση εντολής και τι είναι εξάρτηση δεδομένων; Να αναφερθούν τα είδη εξάρτησης δεδομένων.

Χρονοπρογραμματισμός κώδικα (ακολουθιακός ή ASAP)

Η σχέση $eda = MAX((0.875 * x + 0.5 * y), x)$ όπου $x = MAX(|a|, |b|)$ και $y = MIN(|a|, |b|)$ αποτελεί μία προσέγγιση της ευκλείδειας απόστασης στο επίπεδο, η ακριβής τιμή της οποίας δίνεται από την έκφραση $\sqrt{a^2 + b^2}$. Ζητείται ο σχεδιασμός αρχικά του γράφου ροής δεδομένων που υπολογίζει τη μεταβλητή eda και στη συνέχεια το χρονοπρόγραμμα που προκύπτει με την τεχνική ASAP (As Soon As Possible). Έχετε στη διάθεσή σας τις εξής μονάδες υλικού: αθροιστές (ADD), αφαιρέτες (SUB), εξαγωγείς απόλυτης τιμής (ABS), εξαγωγείς ελαχίστου (MIN) και μεγίστου (MAX), καθώς και αριστερούς (SHL) και δεξιούς (SHR) λογικούς ολισθητές κατά σταθερή ποσότητα n . Όλες οι μονάδες υλικού απαιτούν 1 κύκλο καθυστέρησης.



Άλλα θέματα στο χρονοπρογραμματισμό κώδικα

- 1) Ζητείται να σχεδιαστεί ο γράφος ροής δεδομένων για τον πολλαπλασιασμό μιας εισόδου x με τις σταθερές 5, 17 και 23. Στη συνέχεια να δοθεί το αντίστοιχο χρονοπρόγραμμα που προκύπτει με ακολουθιακή δρομολόγηση ή με δρομολόγηση ASAP και ο αριθμός των απαιτούμενων κύκλων μηχανής για την εκτέλεσή του. Η μονάδα θα διαθέτει τις αντίστοιχες εξόδους u, v, w . Έχετε στη διάθεσή σας τις εξής μονάδες υλικού: αθροιστές (ADD), αφαιρέτες (SUB), και αριστερούς (SHL) και δεξιούς (SHR) λογικούς ολισθητές κατά σταθερή ποσότητα n . Όλες οι μονάδες υλικού απαιτούν 1 κύκλο μηχανής. Σημειώνεται ότι η ολίσθηση κατά n θέσεις αριστερά ισοδυναμεί με πολλαπλασιασμό με το 2^n και η ολίσθηση κατά n θέσεις δεξιά, με διαίρεση με το 2^n .

Ασκήσεις προς επίλυση στο χρονοπρογραμματισμό κώδικα

- Έστω η υποθετική αρχιτεκτονική RISC:

Διαμόρφωση	Συμπεριφορά	Κύκλοι μηχανής
ADD/SUB R1, R2, R3	$R1 := R2 \pm R3$	1
MUL R1, R2, R3	$R1 := R2 \times R3$	2
DIV R1, R2, R3	$R1 := R2 / R3$	4
LOAD R1, imm(R2)	$R1 := \text{MEM}(R2 + \text{imm})$	1
STORE imm(R2), R1	$\text{MEM}(R2 + \text{imm}) := R1$	2

- 1 Ζητείται ο χρονοπρογραμματισμός ASAP για τους υπολογισμούς $Q = (A * B) - (C/D)$ και $Y = (B - C) - (D * E)$ ή
- 2 Ζητείται ο ακολουθιακός/ASAP χρονοπρογραμματισμός του εξής κώδικα

```
LOAD R1, C
LOAD R2, D
LOAD R3, B
LOAD R4, A
DIV R5, R1, R2
MUL R6, R3, R4
SUB R5, R6, R5
STORE X, R5
LOAD R5, E
MUL R5, R5, R2
SUB R6, R3, R1
SUB R6, R6, R5
STORE Y, R6
```

Ερώτηση κρίσεως/σύνθεσης γνώσεων: Τι είναι ο βελτιστοποιητής χαμηλού επιπέδου;

- Ο βελτιστοποιητής χαμηλού επιπέδου χρησιμοποιείται ορισμένες φορές για περαιτέρω βελτίωση του τελικού κώδικα
- Αξιοποιεί ιδιαίτερα χαρακτηριστικά της στοχευόμενης αρχιτεκτονικής
- Παραδείγματα βελτιστοποιητών χαμηλού επιπέδου
 - Χρονοπρογραμματιστής εντολών (instruction scheduler): τοποθετεί τις εντολές του επεξεργαστή σε χρονοθυρίδες (time-slots) για την παράλληλη εκτέλεσή τους
 - Υπερβελτιστοποιητής (superoptimizer): βελτιστοποιεί περιοχές του τελικού κώδικα με εφαρμογή ωμής δύναμης
 - Βελτιστοποιητής κλειδαρότρυπας (peephole optimizer): επιβάλλει μακρο-αντικαταστάσεις με ή χωρίς συνθήκη, εξετάζοντας κάθε φορά ένα 'παράθυρο' του τελικού κώδικα

Δ7: Βελτιστοποιήσεις για εκμετάλλευση της παραλληλίας και ανάδειξη της τοπικότητας

- Η διαδικασία της βελτιστοποίησης
- Γενικευμένη δομή βρόχων και πεδίο επανάληψης
- Loop unswitching, loop reversal
- Strip mining
- Loop tiling
- Loop unrolling
- Software pipelining και σύγκριση με loop unrolling

Ενδεικτικά θέματα από τη βελτιστοποίηση βρόχων

- 1) Να εφαρμοστεί πλακόστρωση βρόχων (loop tiling) για μέγεθος πλακιδίου ίσο με 16. Οι πίνακες a, b έχουν από n στοιχεία.

```
for (i = 0; i < n-1; i++) {  
    b[i] += (a[i] + a[i+1])/2;  
}
```

- β) Να εφαρμοστεί loop unswitching και loop unrolling (με unroll factor $u = 4$) στο παρακάτω τμήμα κώδικα. Οι πίνακες a, b, x έχουν από 100 στοιχεία.

```
for (i = 0; i < 100; i = i + 1) {  
    if (c > 10) {  
        x[i] = a[i] + b[i];  
    } else {  
        x[i] = a[i] - b[i];  
    }  
}
```

Δ8: Γέννηση τελικού κώδικα για RISC επεξεργαστές

- Γενικά στοιχεία για την αρχιτεκτονική επεξεργαστή MIPS
- Γενικό μοντέλο κλήσης υπορουτινών
- Κλήση υπορουτινών με σύνδεση
- Αντίστροφη ερμηνεία (reverse engineering, disassembling) προγραμμάτων MIPS-I, MIPS32
- Γέννηση ευθύγραμμου κώδικα και κώδικα με φυσικούς βρόχους

Κώδικας συμβολομεταφραστή για τον MIPS: Εύρεση μεγαλύτερου στοιχείου πίνακα

```
                                # a0: address of element mem[0]
                                # a1: value to match
prog:
  add  $t0, $zero, $zero # t0 = 0
  add  $v0, $zero, $zero # v0 = 0
  add  $v1, $zero, $zero # v1 = 0
loop:
  sltu $t2, $t0, $a1      # t2 = (t0 < a1)
  beq  $t2, $zero, fin    # if (!t2) goto fin
  lw   $t1, 0($a0)        # t1 = mem[a0]
  sltu $t2, $t1, $v0      # t2 = (t1 < v0) UNSIGNED!
  bne  $t2, $zero, skip   # if (t2) goto skip
  add  $v0, $t1, $zero    # v0 = t1
  add  $v1, $t0, $zero    # v1 = t0
skip:
  addi $t0, $t0, 1        # t0 = t0 + 1
  addi $a0, $a0, 4        # a0 = a0 + 4
  j    loop               # goto loop
fin:
                                # $finish
```

Το πρόγραμμα αυτό βρίσκει το μεγαλύτερο στοιχείο σε έναν πίνακα απρόσημων 32-bit ακεραίων και επιστρέφει το στοιχείο αυτό στον καταχωρητή \$v0 και τη θέση του στον πίνακα στον \$v1