Προηγμένα Θέματα Θεωρητικής Πληροφορικής

Επαναστοχεύσιμοι μεταγλωττιστές

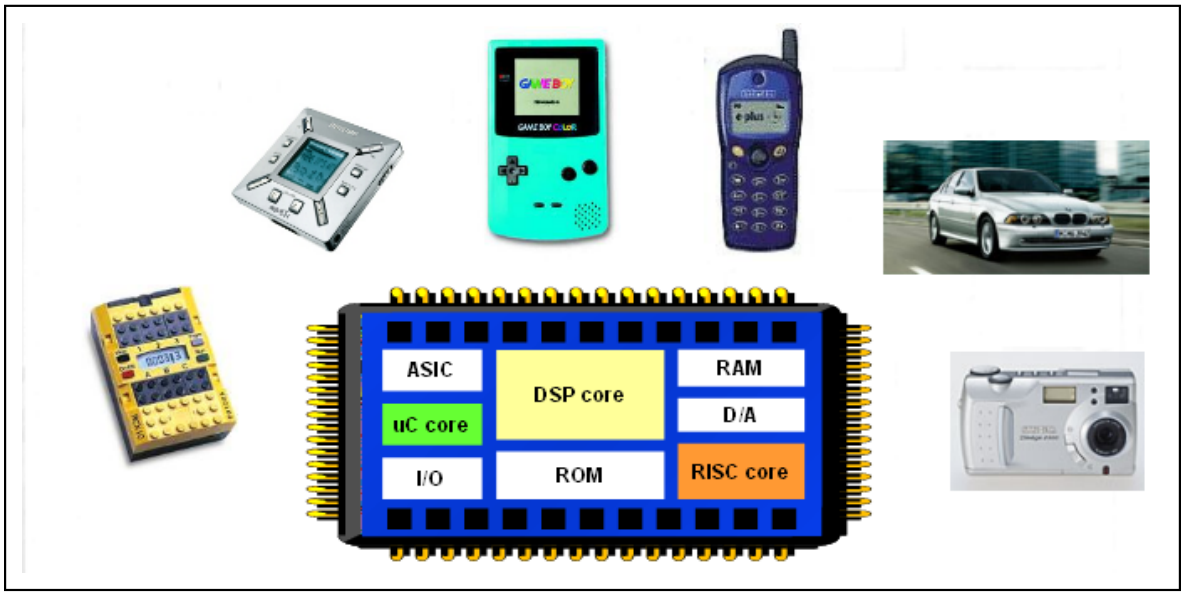
Νικόλαος Καββαδίας
nkavv@uop.gr

15 Ιουνίου 2010

Σκιαγράφηση της διάλεξης

- Εισαγωγή στα ενσωματωμένα συστήματα (embedded systems)
- Η χρησιμότητα των επαναστοχεύσιμων μεταγλωττιστών
- Χαρακτηριστικά των επαναστοχεύσιμων μεταγλωττιστών
- Παρουσίαση των δημοφιλέστερων περιπτώσεων:
  - GCC
  - LLVM
  - COINS
  - LCC
  - PCC

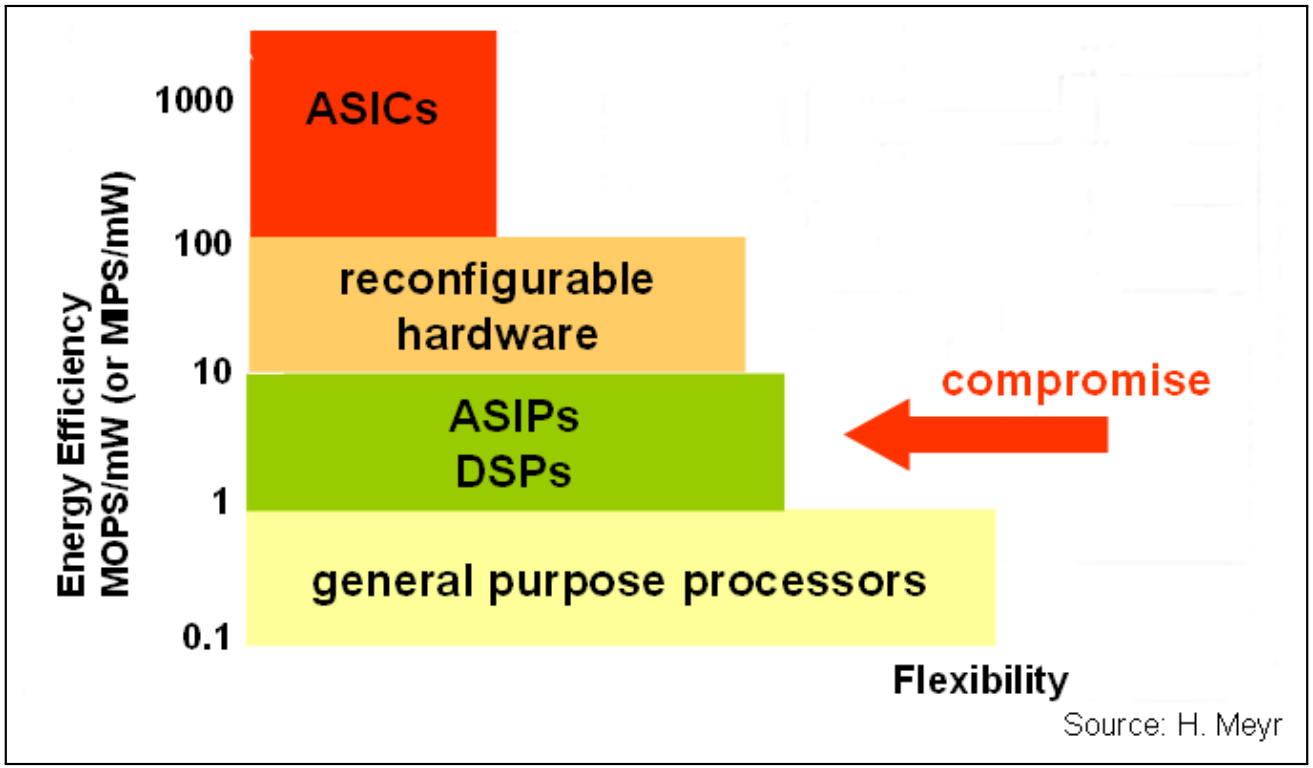
Η αναγκαιότητα των (επαναστοχεύσιμων) μεταγλωττιστών για ενσωματωμένους επεξεργαστές



- Ενσωματωμένο σύστημα: οποιοδήποτε σύστημα επεξεργασίας διαθέτει περιορισμένη διεπαφή με το χρήστη
- Συνήθως 'κρυμμένο': αυτοκίνητα, σπιτικές ηλεκτρονικές συσκευές, ιατρικά όργανα, κινητά τηλέφωνα, παιχνιδομηχανές κ.λ.π.
- Η αυξανόμενη πολυπλοκότητα των ενσωματωμένων συστημάτων, οδήγησε στη χρήση προγραμματιζόμενων επεξεργαστών
- Για τη διευκόλυνση του προγραμματισμού τους είναι αναγκαία η χρήση μεταγλωττιστών

Κατηγοριοποίηση των ενσωματωμένων επεξεργαστών

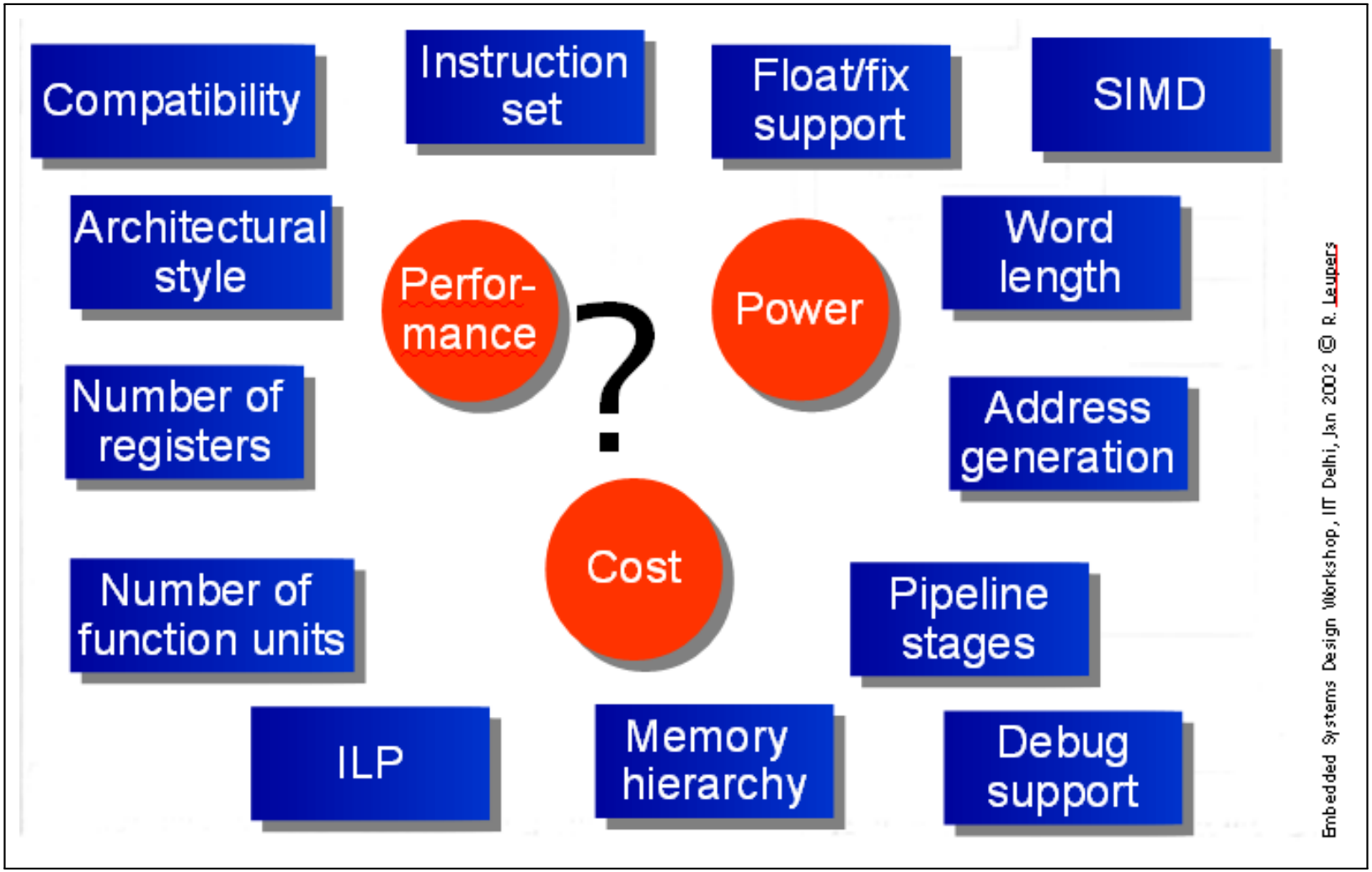
Κύρια χαρακτηριστικά ενός ενσωματωμένου επεξεργαστή είναι η επάρκεια (ταχύτητα επεξεργασίας, κατανάλωση ισχύος/ενέργειας, επιφάνεια ολοκληρωμένου) σε σχέση με την ευελιξία του (δυνατότητα προγραμματισμού, ευκολία προσαρμογής σε μεταβαλλόμενα δεδομένα από το περιβάλλον του)



Ενσωματωμένοι επεξεργαστές και τα αρχιτεκτονικά χαρακτηριστικά τους

1. Είναι σχεδιασμένοι με κύριο στόχο την επάρκεια (υψηλή ταχύτητα επεξεργασίας με χαμηλή κατανάλωση ισχύος/ενέργειας)
2. Ευκολία/αμεσότητα του προγραμματισμού τους
3. Μεγάλη ποικιλία ενσωματωμένων επεξεργαστών στη σημερινή αγορά
4. Αρχιτεκτονική Harvard (ξεχωριστή αποθήκευση για δεδομένα και εντολές)
5. Ετερογενής αρχιτεκτονική καταχωρητών
6. Υλοποιούμενη αριθμητική: saturating arithmetic, αριθμητική σταθερής υποδιαστολής
7. Εξειδικευμένες εντολές όπως για διευθυνσιοδότηση bit, πολλαπλασιασμό-και-συσσώρευση
8. Συνήθως απουσιάζουν: MMU, superscalar χαρακτηριστικά, αριθμητική κινητής υποδιαστολής

Επιμέρους παράμετροι του σχεδιασμού ενσωματωμένων επεξεργαστών



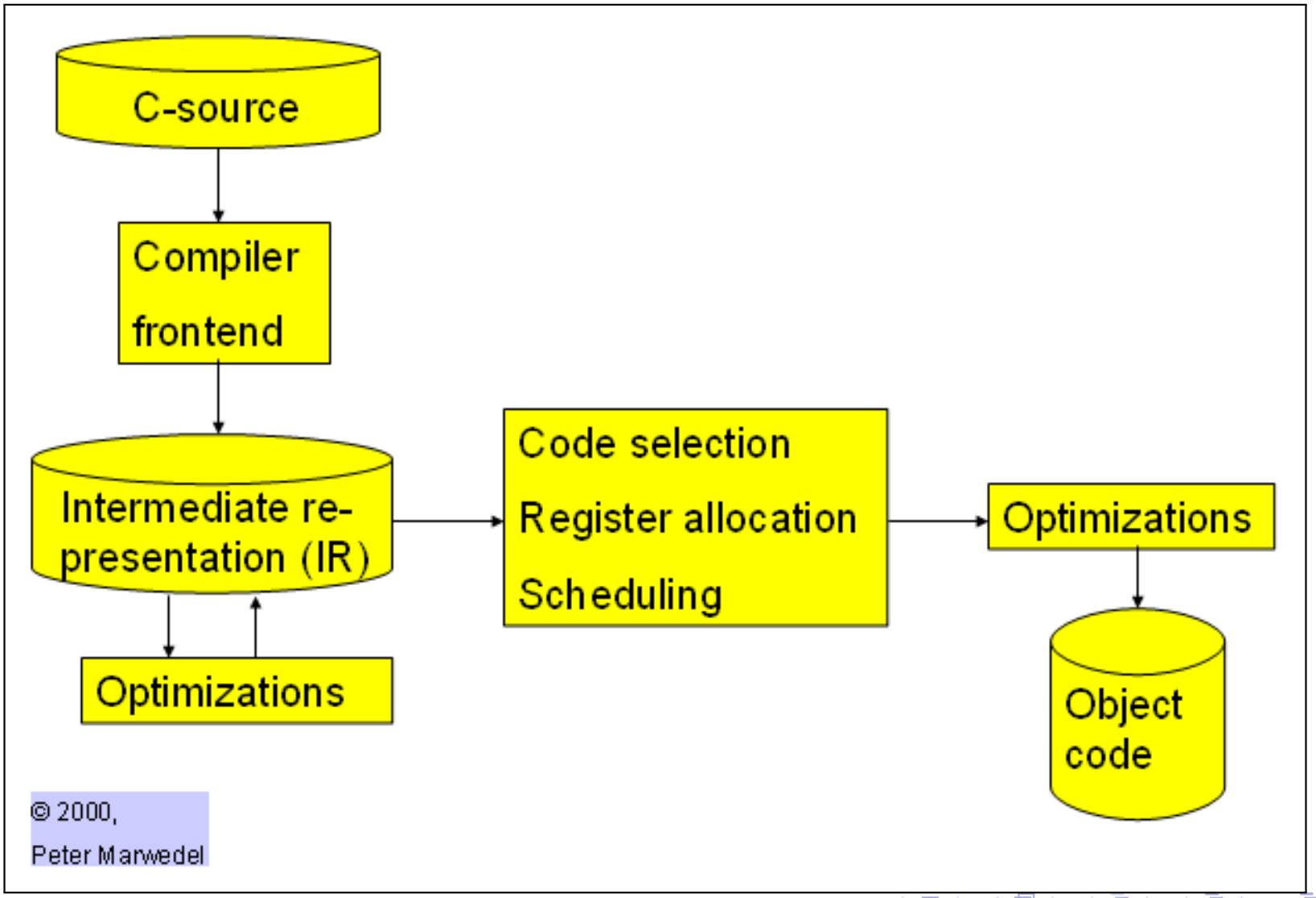
Ο κόσμος της ANSI/ISO C

- Η πιο διαδεδομένη γλώσσα για τη δημιουργία λογισμικού εφαρμογών και συστημάτων
- Η γλώσσα ISO C διέπεται από το πρότυπο ISO/IEC 9899:1999
- Σε κοινή χρήση και το 'παλαιό' πρότυπο (C89)
- Προτάσεις για την άρση των περιορισμών της C και την επέκταση της χρήσης της σε άλλα πεδία (όπως η σύνθεση υλικού υψηλού επιπέδου)
  - Valen-C (Yasuura, Kyushu-University at Fukuoka)
  - DSP-C (proposed by ACE (www.ace.nl))
  - SpecC (UC Irvine) by Daniel Gajski
  - SystemC: Hardware data types and simulation (`http://www.systemc.org`)
  - HardwareC
  - Embedded C++ (EC++)
  - και άλλες επεκτάσεις

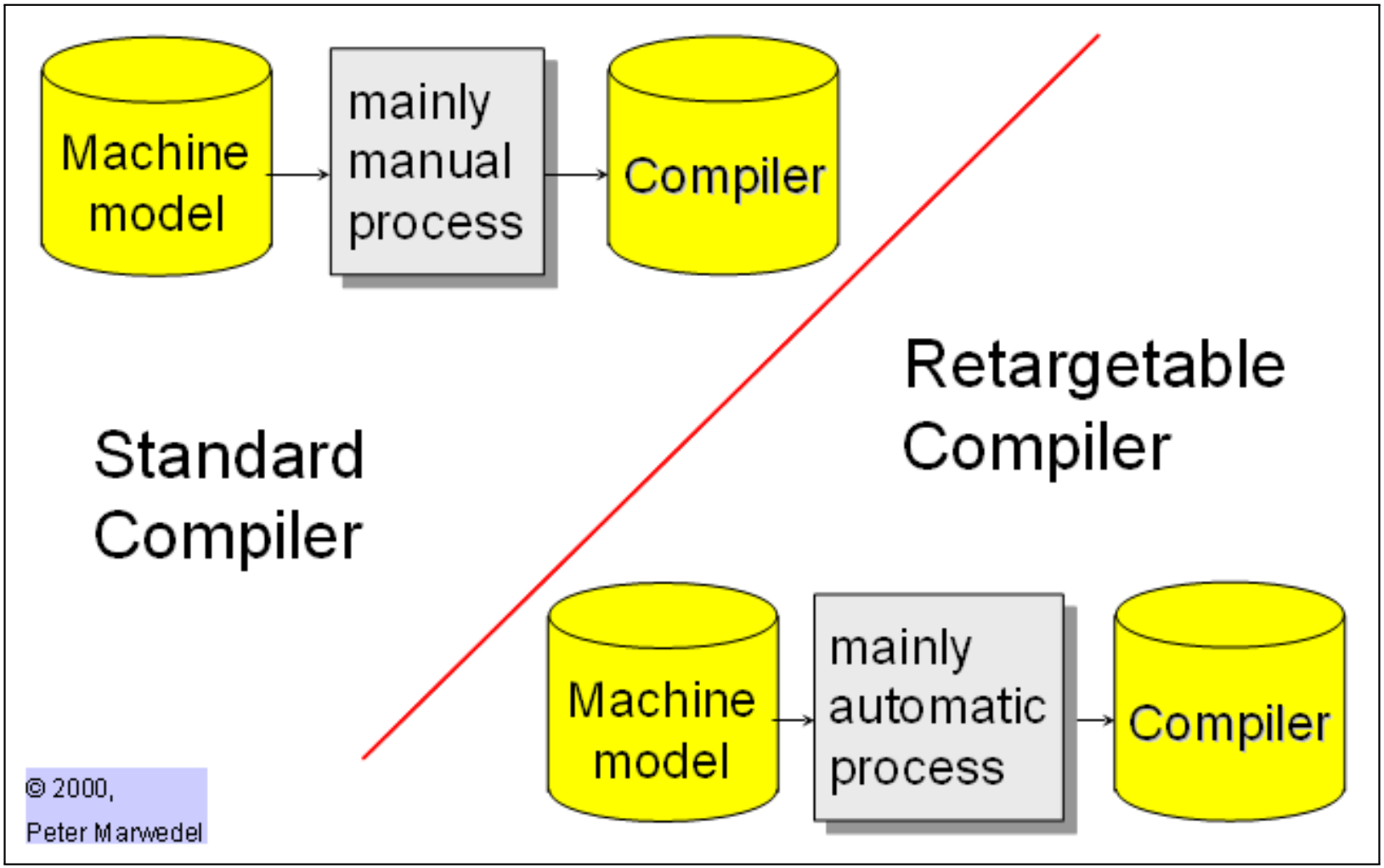
Υποδομή για την ανάπτυξη και σχεδιασμό ενσωματωμένων επεξεργαστών

- Επαναστοχεύσιμοι μεταγλωττιστές (retargetable compilers)
- Προσομοιωτές (simulators)
  - ακρίβειας εντολής: στα πρώτα στάδια εκσφαλμάτωσης, ανάπτυξη εξομοιωτών (emulators) συστημάτων
  - ακρίβειας κύκλου: λεπτομερές μικροαρχιτεκτονικό μοντέλο, καταγράφει την κατάσταση (state) του επεξεργαστή ανά κύκλο μηχανής
- Λογισμικά εργαλεία ανάπτυξης εφαρμογών (software development tools)
  - γεννήτορας επιλογέα κώδικα, συμβολομεταφραστής (`as`), συνδέτης (`ld`), αποσυμβολομεταφραστής (`objdump`), αποσφαλματωτής (`gdb`), φορτωτής
- Άλλα εργαλεία (λογισμικές βιβλιοθήκες: δομών δεδομένων, αλγορίθμων, απεικόνισης και ταυτοποίησης γράφων, κ.α.)

Τυπική οργάνωση του μεταγλωττιστή



Ο επαναστοχεύσιμος μεταγλωττιστής σε σχέση με τον τυπικό μεταγλωττιστή



Χαρακτηριστικά των επαναστοχεύσιμων μεταγλωττιστών (1)

- Κριτήρια καταλληλότητάς τους για τη χρήση στο σχεδιασμό και προγραμματισμό ενσωματωμένων επεξεργαστών
  - Το είδος της ενδιάμεσης αναπαράστασης (IR)
  - Επεκτασιμότητα (ευκολία στην προσθήκη/τροποποίηση backend, περάσματα ανάλυσης ή/και βελτιστοποίησης)
  - Δυνατότητα επαναστόχευσης από machine description
  - Τεκμηρίωση, παραδείγματα χρήσης, βάση χρηστών, ανάπτυξη και συντήρηση
- Σήμερα (2010) πάνω από 30 επαναστοχεύσιμοι μεταγλωττιστές ανοικτού κώδικα για γλώσσες υψηλού επιπέδου (όπως C, Java) διατίθενται χωρίς κόστος

Χαρακτηριστικά των επαναστοχεύσιμων μεταγλωττιστών (2)

- Δημοφιλείς μεταγλωττιστές
  - GCC: δεκάδες αρχιτεκτονικές ξενιστή/στόχου, ισχυρές βελτιστοποιήσεις, δύσκολη η συγγραφή νέου backend, άναρχα δομημένος
  - LLVM: μοντέρνα σχεδίαση, υποστήριξη από Apple Inc.
  - COINS: γραμμένος σε Java, ισχυρές βελτιστοποιήσεις για παραλληλοποίηση
  - Phoenix: πρωτότυπο περιβάλλον από Microsoft
  - PCC: αναβίωση του Portable C Compiler
  - LCC: λιτός, κατάλληλος μόνο για απλούς RISC, χωρίς επεκτάσιμη υποδομή
  - SUIF/MachSUIF: καθαρό API, επεκτάσιμος, δύσκολη η προσθήκη νέων backend, χρήσιμος στην έρευνα
  - Άλλοι μεταγλωττιστές:
    - Trimaran, ACK, SPAM
  - Εμπορικοί μεταγλωττιστές: CoSy ACE, Archelon

GCC (1)

- Ο πιο διαδεδομένος επαναστοχεύσιμος μεταγλωττιστής
- Χρησιμοποιείται ως ο μεταγλωττιστής συστήματος σε λειτουργικά συστήματα Unix/Linux
- Μπορεί να χρησιμοποιηθεί για τη γέννηση κώδικα για τρίτους επεξεργαστές (ενώ λειτουργεί σε άλλο επεξεργαστή): cross compiler
- Υποστηρίζει τις γλώσσες C/C++, Objective-C, Java, Fortran, Ada
- Μεταφερτός σε νέες αρχιτεκτονικές μέσω αρχείου περιγραφής μηχανής και βοηθητικών συναρτήσεων στη C
- Πρόσφατες προσθήκες στον GCC:
  - Από την έκδοση 4.0: υποστήριξη SSA
  - Ενδιάμεση αναπαράσταση GIMPLE
  - Επικοινωνία με εξωτερικά plug-in

ℹ Η ανάπτυξή του, ορισμένες φορές, περιορίζεται από θέματα πολιτικής του σχεδιασμού του (π.χ. ελευθερία λογισμικού)

GCC (2)

- Παράδειγμα από περιγραφή μηχανής

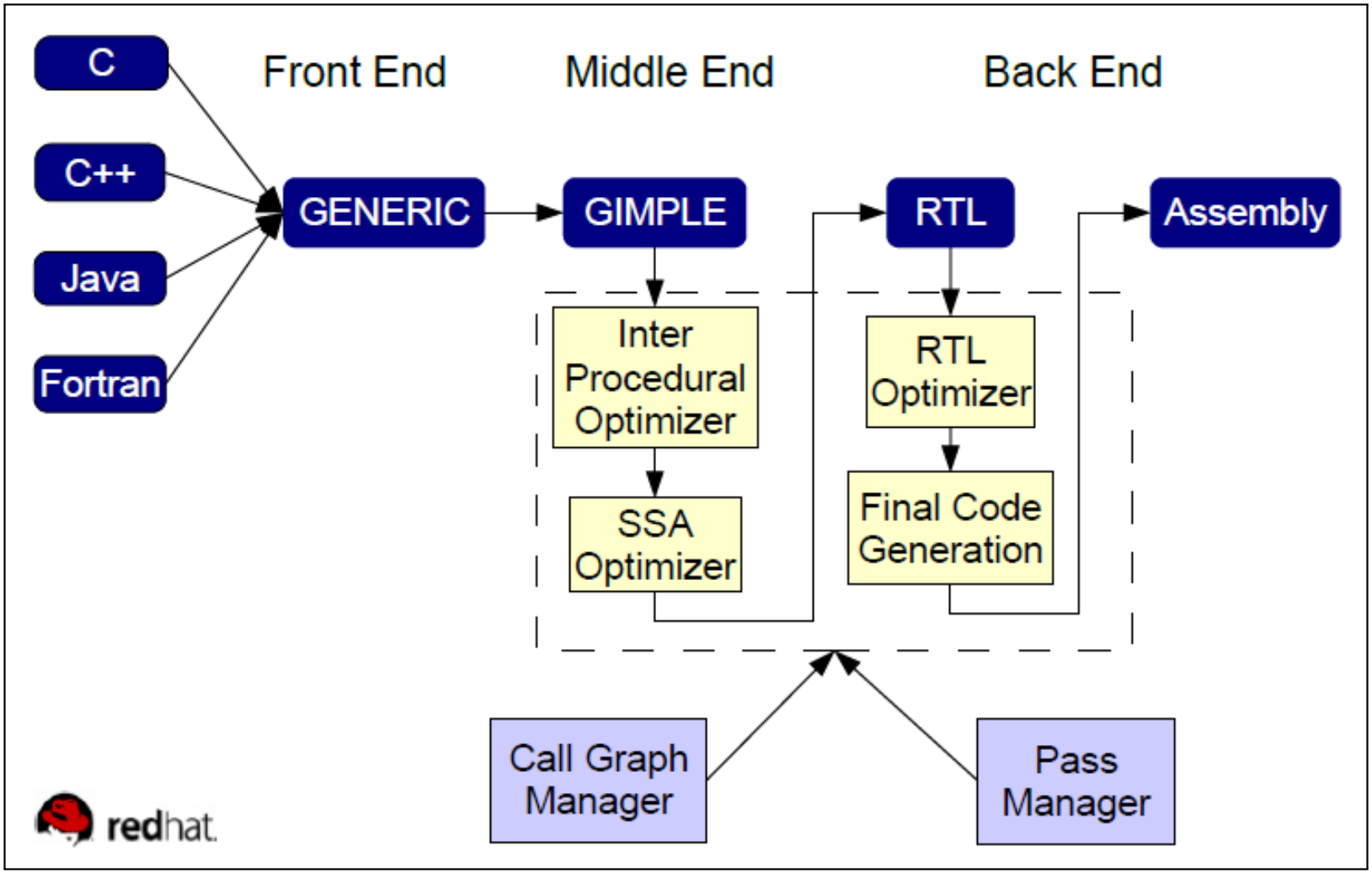
```
(define_insn "subsf3"
  [(set (match_operand:SF 0 "register_operand" "=f")
        (minus:SF (match_operand:SF 1 "register_operand" "f")
                  (match_operand:SF 2 "register_operand" "f")))]
  ""
  "subf\\t%0,%1,%2")
```

- ☞ Δυσκολία στην περιγραφή μη κανονικών αρχιτεκτονικών
- Στόχοι του GCC

  The main goal of GCC was to make a good, fast compiler for machines in the class that the GNU system aims to run on: 32-bit machines that address 8-bit bytes and have several general registers.

  Elegance, theoretical power and simplicity are only secondary.

Η εσωτερική οργάνωση του GCC



Μετατροπή σε μορφή SSA

Original program

```
a = 3;
b = 1;
while (b <= 123) {
  a = a + b;
  if (a <= 42)
    a = a + 7;
  b = b + 5;
}
```

→

Tree-SSA form

```
a = 3;
b = 1;
loop
  c = ϕ (a, f);
  d = ϕ (b, g);
  if (d > 123) goto end;
  if (c > 42) goto else;
    e = c + 7;
    goto endif;
else:
endif:
  h = ϕ (e, d);
  f = h + d;
  g = d + 5;
endloop
end:
```

Renaming:

a ⟶ a, c, e, f

b ⟶ d, g

Μετατροπή από μορφή SSA σε κώδικα τύπου C

Tree-SSA form

```
a = 3;
b = 1;
loop
  c = ϕ (a, f);
  d = ϕ (b, g);
  if (d > 123) goto end;
  if (c > 42) goto else;
    e = c + 7;
    goto endif;
else:
endif:
  h = ϕ (e, d);
  f = h + d;
  g = d + 5;
endloop
end:
```

→

Imperative program

```
a = 3;
b = 1;
c = a; d = b; // copy
loop
  if (d > 123) goto end;
  if (c > 42) goto else;
    e = c + 7;
    h = e; // copy
    goto endif;
else:
    h = d; // copy
endif:
  f = h + d;
  g = d + 5;
  c = f; d = g; // copy
endloop
end:
```

Απλό παράδειγμα γέννησης κώδικα για εντολή φόρτωσης σταθεράς

```
(define_insn
   "movsi"
    (set
      (match_operand 0 "register_operand" "r")
      (match_operand 1 "const_int_operand" "k")
      )
   "" /*  C boolean expression, if required */
   "li %0, %1"
)
```

Development

D.1283 = 10; ⇒

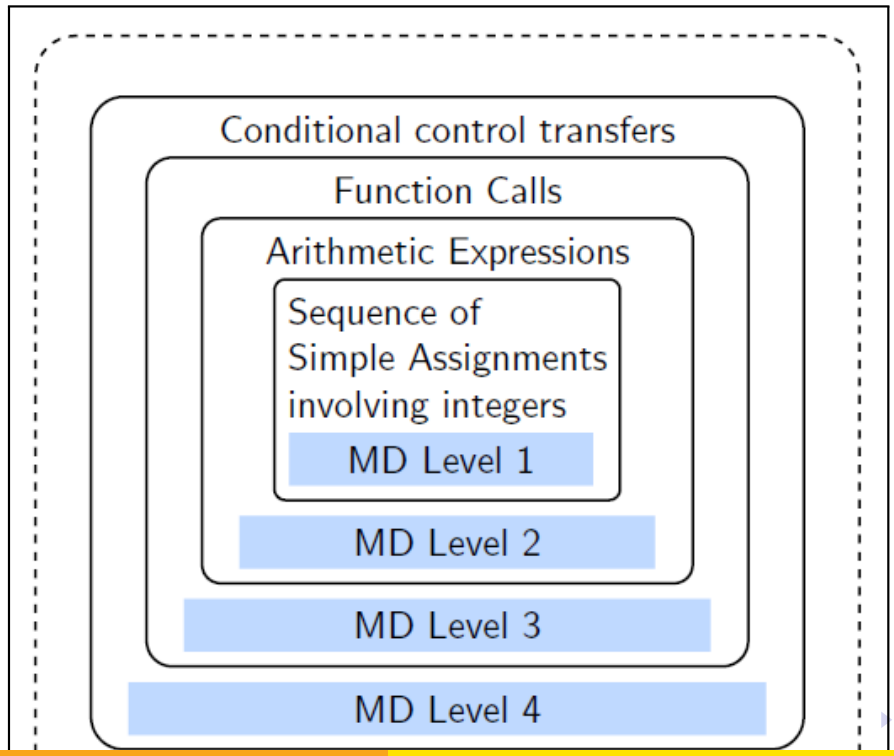
```
(set
    (reg:SI 58 [D.1283])
    (const_int 10:  [0xa])
)
```

⇒ li $t0, 10

Use

Αυξητική κατασκευή περιγραφής μηχανής για τον GCC (Uday Khedker)

- Η κατασκευή ενός νέου backend στον GCC είναι επίπονη:
  - Υλοποίηση μικρών, ελεγχόμενων βημάτων
  - Σε κάθε βήμα προστίθενται νέα γραμματικά στοιχεία της πηγαίας γλώσσας
  - Σε αντίθεση με την κοινή άποψη ότι αυτό που προσδιορίζεται αυξητικά είναι η αρχιτεκτονική-στόχος



Υποστηριζόμενες αρχιτεκτονικές επεξεργαστή από τον GCC

- GCC target processor families as of version 4.3 include:
  - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU (Cell), System/390/zSeries, SuperH, SPARC, VAX
- Lesser-known target processors supported in the standard release have included:
  - A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MIL-STD-1750A, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
- Additional processors have been supported by GCC versions maintained separately from the FSF version:
  - Cortus APS3, D10V, eSi-RISC, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, OpenRISC 1200, PDP-10, TIGCC (m68k variant), System/370, Z8000, PIC24/dsPIC, NEC SX

Παραπομπές για τον GCC

- GCC on the Free Software Foundation web site
  `http://gcc.gnu.org`
- The official GCC manuals and user documentation, by the GCC developers
  `http://gcc.gnu.org/onlinedocs/`
- GCC Wiki | Tutorial and Optimization Course
  `http://gcc.gnu.org/wiki/OptimizationCourse`
- HiPEAC GCC Tutorials
  `http://www.hipeac.net/gcc-tutorial`
- Collection of GCC 4.0.2 architecture and internals documents
  `http://www.cse.iitb.ac.in/grc/docs.html`
- Essential abstractions in GCC workshop (2009)
  `http://www.cse.iitb.ac.in/grc/gcc-workshop-09/`

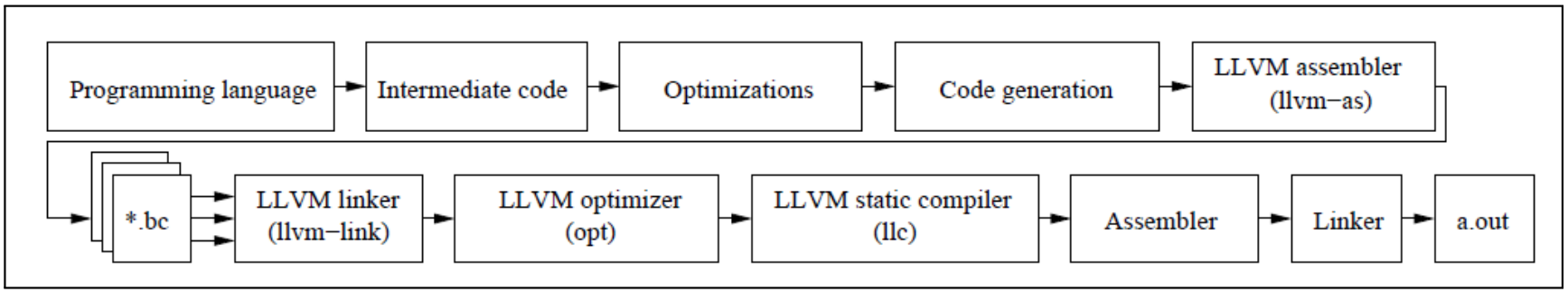
LLVM (Low-Level Virtual Machine

- LLVM: Μοντέρνος επαναστοχεύσιμος μεταγλωττιστής (`http://llvm.org`)
- LLVM bytecode/bitcode: ενδιάμεση αναπαράσταση και εικονική μηχανή
- Αρχική ιδέα από τον Vikram Adve
- Σχεδιασμός από τον Chris Lattner
- Έμπρακτη υποστήριξη από την Apple Inc.
- Το frontend clang προσφέρει απευθείας μεταγλώττιση από C/C++/Objective-C σε LLVM bytecode (`http://clang.llvm.org`)
- Σχετικά εύκολα επαναστοχεύσιμος
- Διαχειρίσιμη η επαναστόχευσή του σε επεξεργαστές διαφορετικών κατηγοριών
- Πολύ καλή τεκμηρίωση

Χαρακτηριστικά του LLVM

- Έχει συνταχθεί σε καλογραμμένη C++
- Όλοι οι μηχανισμοί έχουν υλοποιηθεί σε ανεξάρτητα εφαρμόσιμα περάσματα (passes)
- Pluggable μετασχηματισμοί, αναλύσεις, καταμεριστές καταχωρητών, κ.λ.π.
- Βοηθητικά εργαλεία
  - llc - invoke static back-ends
  - lli - bitcode interpreter, use JIT
  - bugpoint - reduce code from crashes
  - opt - run optimizations on bitcodes
  - llvm-extract - extract/delete functions and data
  - llvm-dis, llvm-as, llvm-ld: binutils-like tools
- Υποστηριζόμενες αρχιτεκτονικές (επεξεργαστή ή εικονικές)
  - Alpha, ARM, C, CellSPU, IA64, Microblaze, MIPS, MSIL, PowerPC, SPARC, x86, x86_64, XCore, PIC-16

Η διαδικασία της μεταγλώττισης στον LLVM



Παράδειγμα 1: Μεταγλώττιση απλού βρόχου

Χρήση του LLVM από τη γραμμή εντολών

```
${LLVM_GCC_PATH}/llvm-gcc -O3 -emit-llvm loop1.c -S -c -o loop1.bc
```

Κώδικας ANSI C για το βρόχο

```
void loop1(int inp, int *outp) {
  int ix;
  for (ix = 0; ix < inp; ix++) ;
  *outp = ix;
}
```

LLVM bitcode

```
; ModuleID = 'loop1.c'
target datalayout = "e-p:32:32:32-i1:8:8-i8:8:8-i16:16:16-i32:
32:32-i64:64:64-f32:32:32-f64:64:64-v64:64:64-v128:128:128-a0:0:64-f80:32:32-n8:16:32"
target triple = "i386-mingw32"

define void @loop1(i32 %inp, i32* nocapture %outp) nounwind {
entry:
  %0 = icmp sgt i32 %inp, 0                       ; <i1> [#uses=1]
  %ix.0.lcssa = select i1 %0, i32 %inp, i32 0     ; <i32> [#uses=1]
  store i32 %ix.0.lcssa, i32* %outp, align 4
  ret void}
```

Παράδειγμα 2: Εφαρμογή απαρίθμησης πληθυσμού

Κώδικας ANSI C

```
void popcount(int inp, int *outp) {
  int data = inp, count = 0, temp;
  while (data != 0) {
    count = count + (data & 0x1);
    data = data >> 0x1;
  }
  *outp = count;
}
```

LLVM bitcode

```
define void @popcount(i32 %inp, i32* nocapture %outp) nounwind {
entry:
  %0 = icmp eq i32 %inp, 0                        ; <i1> [#uses=1]
  br i1 %0, label %bb2, label %bb
bb:                                               ; preds = %entry, %bb
  %count.04 = phi i32 [ %2, %bb ], [ 0, %entry ]  ; <i32> [#uses=1]
  %data.03 = phi i32 [ %3, %bb ], [ %inp, %entry ] ; <i32> [#uses=2]
  %1 = and i32 %data.03, 1                        ; <i32> [#uses=1]
  %2 = add nsw i32 %1, %count.04                  ; <i32> [#uses=2]
  %3 = ashr i32 %data.03, 1                       ; <i32> [#uses=2]
  %4 = icmp eq i32 %3, 0                          ; <i1> [#uses=1]
  br i1 %4, label %bb2, label %bb
bb2:                                              ; preds = %bb, %entry
  %count.0.lcssa = phi i32 [ 0, %entry ], [ %2, %bb ] ; <i32> [#uses=1]
  store i32 %count.0.lcssa, i32* %outp, align 4
  ret void}
```

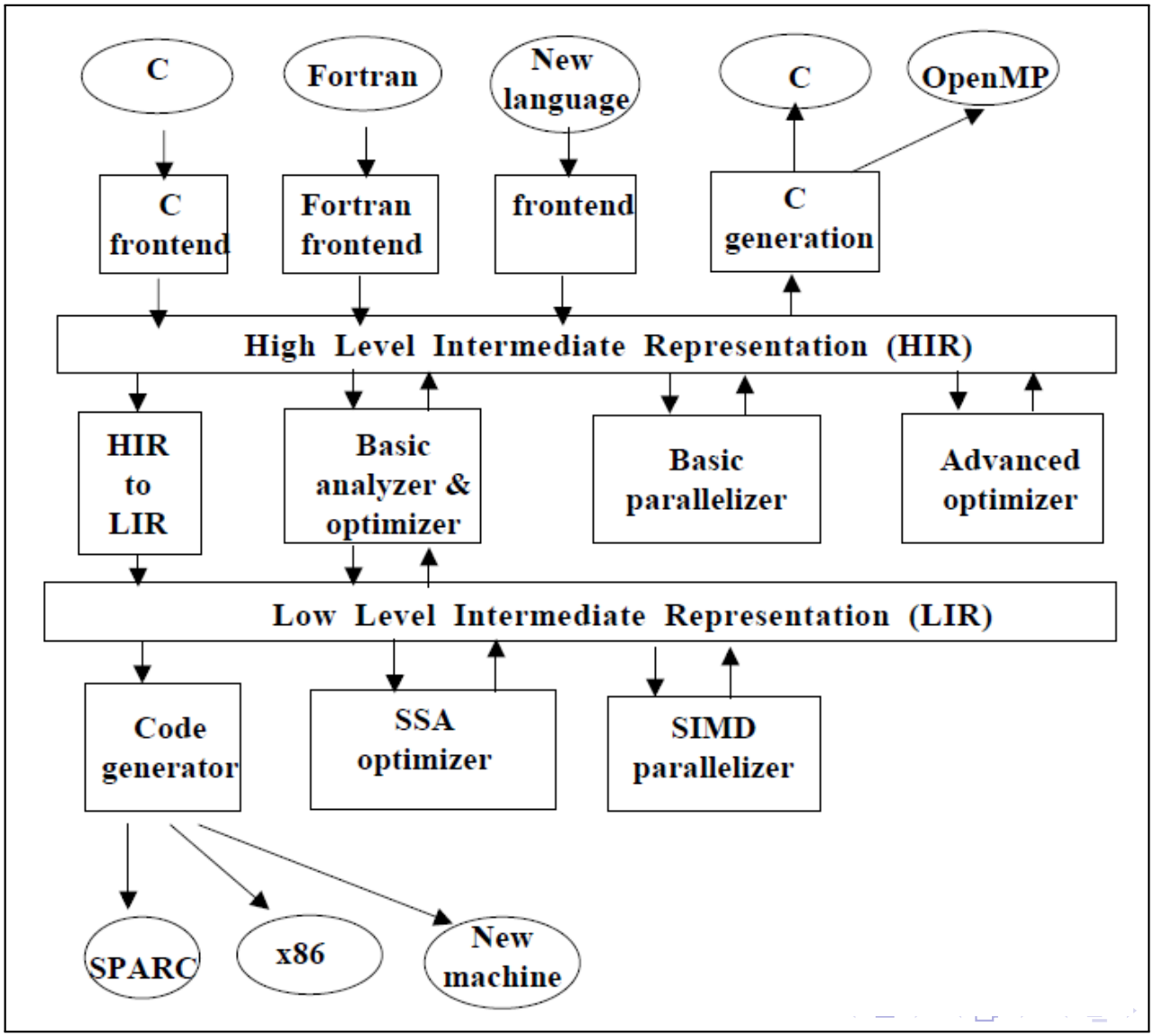
COINS (COmpiler INfraStructure

- Ερευνητικός επαναστοχεύσιμος μεταγλωττιστής (Hosei Univ., Tokyo Institute of Technology, SONY Corporation, Mitsubishi Research Institute), κ.α.
- Ιστοσελίδα: `http://www.coins-project.org/international/`
- Συνταγμένος σε Java
- Υποδομή ανάπτυξης μεταγλωττιστών με σκοπό την έρευνα, την εκπαίδευση και τη χρήση σε περιβάλλον παραγωγής
- Δύο επίπεδα ενδιάμεσης αναπαράστασης: HIR: High-level Intermediate Representation, LIR: Low-level Intermediate Representation
- Πολύ καλή τεκμηρίωση

Χαρακτηριστικά του COINS

- Structure of the COINS compiler
  - Compiler control and driver
  - IR (Intermediate Representation) handler
    - HIR handler, LIR handler, Symbol handler
  - Front-end: C, Fortran
  - Middle-end:
    - Basic and advanced optimizer
    - SSA (Static Single Assignment) optimizer
    - Basic parallelizer, SMP (Symmetric Multi-Processor) parallelizer, SIMD (Single Instruction Multiple Data stream) parallelizer
    - HIR-to-C source code generator
  - Backend
    - Code generator generator (retargetable)
    - Available code generators for: SPARC, x86 (32-bit, 64-bit), ARM, THUMB, MIPS, SH-4, PowerPC, Alpha, Microblaze
    - Instruction scheduler
    - Software pipelining optimizer
    - LIR-to-C source code generator

Η εσωτερική οργάνωση του COINS



Παράδειγμα μεταγλώττισης με τον COINS (1)

Κώδικας ANSI C

```
#include<stdio.h>
int main(){
   int x = 1, y = 2, sum;
   sum = x + y;
   printf("%d\n", sum);
   return 0;
};
```

HIR

```
(prog 1
 <null 0 void>
 <nullNode 0>
 (subpDef 0 void
  <subp 0 <SUBP <> true int> main>
 <null 0 void>
  (labeldSt 2 void
   (list 3
    <labelDef _lab1>)
   (block 5 void
    (assign 6 int
     <var 7 int x>
     <const 8 int 1>)
    (assign 9 int
     <var 10 int y>
     <const 11 int 2>)
    (assign 12 int
     <var 13 int sum>
     (add 14 int
      <var 15 int x>
      <var 16 int y>))
    (expStmt 17 int
     (call 18 int
      (addr 19 <PTR <SUBP <<PTR char> > true int
       <subp 20 <SUBP <<PTR char> > true int> p
      (list 21
       (decay 22 <PTR char>
        <const 23 <VECT 4 0 char> "%d\n">)
       <var 24 int sum>)))
    (return 25 int
     <const 26 int 0>)))))
```

Παράδειγμα μεταγλώττισης με τον COINS (2)

LIR

```
(FUNCTION "main"
  (SYMTAB
    ("functionvalue.5" FRAME I32 4 0)
    ("returnvalue.4" FRAME I32 4 0)
    ("sum.3" FRAME I32 4 0)
    ("y.2" FRAME I32 4 0)
    ("x.1" FRAME I32 4 0) )
    (PROLOGUE (0 0))
    (DEFLABEL (LABEL I32 "_lab1"))
    (SET I32 (MEM I32 (FRAME I32 "x.1")) (INTCONST I32 1))
    (SET I32 (MEM I32 (FRAME I32 "y.2")) (INTCONST I32 2))
    (SET I32 (MEM I32 (FRAME I32 "sum.3"))
      (ADD I32 (MEM I32 (FRAME I32 "x.1"))
      (MEM I32 (FRAME I32 "y.2"))))
    (CALL (STATIC I32 "printf") ((STATIC I32 "string.6")
      (MEM I32 (FRAME I32 "sum.3")))
      ((MEM I32 (FRAME I32 "functionvalue.5"))))
    (SET I32 (MEM I32 (FRAME I32 "returnvalue.4")) (INTCONST I32 0))
    (JUMP (LABEL I32 "_epilogue"))
    (DEFLABEL (LABEL I32 "_epilogue"))
    (EPILOGUE (0 0) (MEM I32 (FRAME I32 "returnvalue.4")))
```

Παράδειγμα μεταγλώττισης με τον COINS (3)

SPARC assembly

```
        .section ".text"
        .align  1
string.6:
        .byte   37
        .byte   100
        .byte   10
        .byte   0
        .align  4
        .global main
main:
        save    %sp,-96,%sp
.L2:
        mov     1,%i0
        mov     2,%i0
        mov     3,%i0
        set     string.6,%o0
        mov     3,%o1
        call    printf
        mov     0,%i0
.L3:
        ret
        restore
```

LCC (1)

- C μεταγλωττιστής από το πανεπιστήμιο του Princeton (Dave R. Hanson, Christopher W. Fraser)
- Ιστοσελίδα: `http://www.cs.princeton.edu/software/lcc`, `http://sites.google.com/site/lccretargetablecompiler/`
- Κώδικας στο δημόσιο πεδίο (public domain) με δέσμευση για μη εμπορική χρήση χωρίς την άδεια των δημιουργών του
- Μικρή βάση κώδικα (λίγα KLOC)
- Λίγες βελτιστοποιήσεις
- Σχετικά εύκολα επαναστοχεύσιμος από ένα αρχείο `.md`
- Μοντέλο επαναστοχεύσιμου μεταγλωττιστή και όχι γεννήτορα μεταγλωττιστή
- Δύσκολη η επαναστόχευσή του σε επεξεργαστές που δεν είναι τύπου RISC ή CISC
- Πολύ καλή τεκμηρίωση

LCC (2)

- AST IR μετατρεπόμενο σε DFTs
- Επιλογή κώδικα με κάλυψη δένδρου
- Τοπικός καταμερισμός καταχωρητών
- Δεν διαθέτει χρονοπρογραμματισμό εντολών

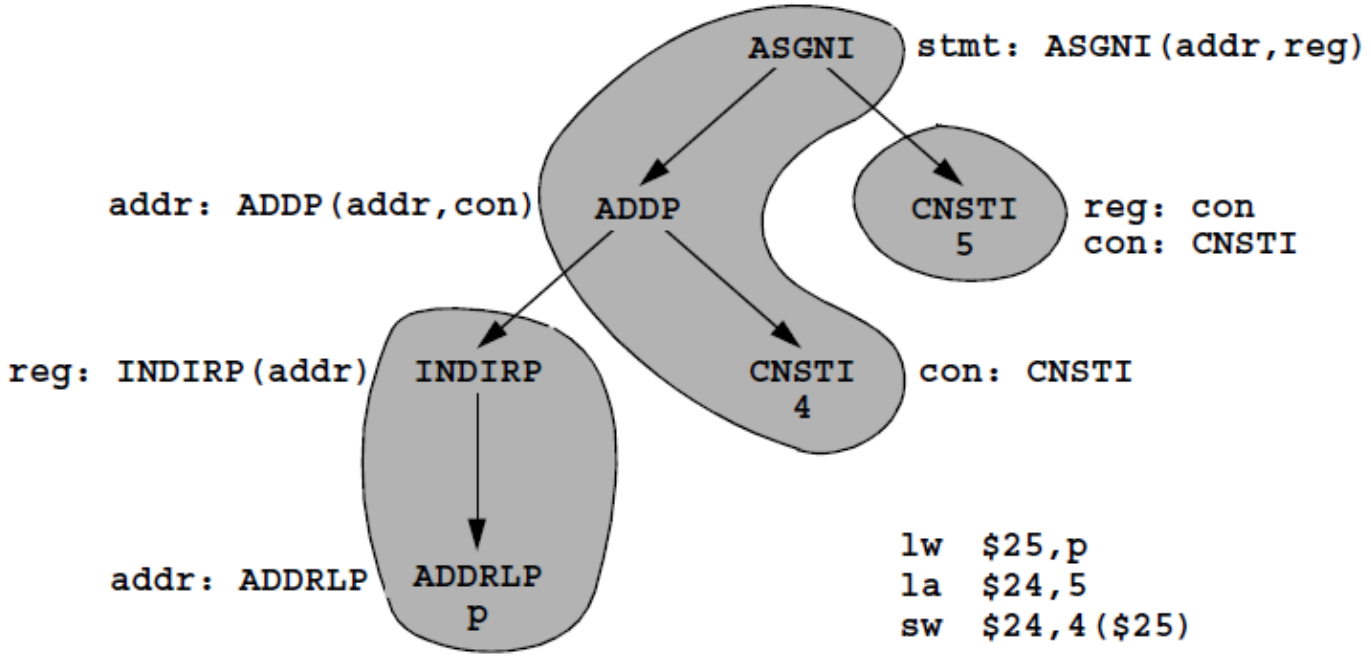
```
int a;
int f(int* p) {
  return a + *p + 1;
}

5. ADDRGP2 a     // address of "a"
4. INDIRI2 #5    // load a
8. ADDRFP2 p     // address of "p"
7. INDIRP2 #8    // load p
6. INDIRI2 #7    // load *p
3. ADDI2 #4 #6   // integer add
9. CNSTI2 1      // constant "1"
2. ADDI2 #3 #9   // integer add
1. RETI2 #2      // integer return
```

Παράδειγμα γέννησης κώδικα για τον MIPS

Generating Code

April 2, 1996

- label tree, e.g., for `int *p; p[1] = 5;`
- pick cheapest rule matching start symbol
- recursively pick rules for the frontier
- fill in and emit corresponding templates

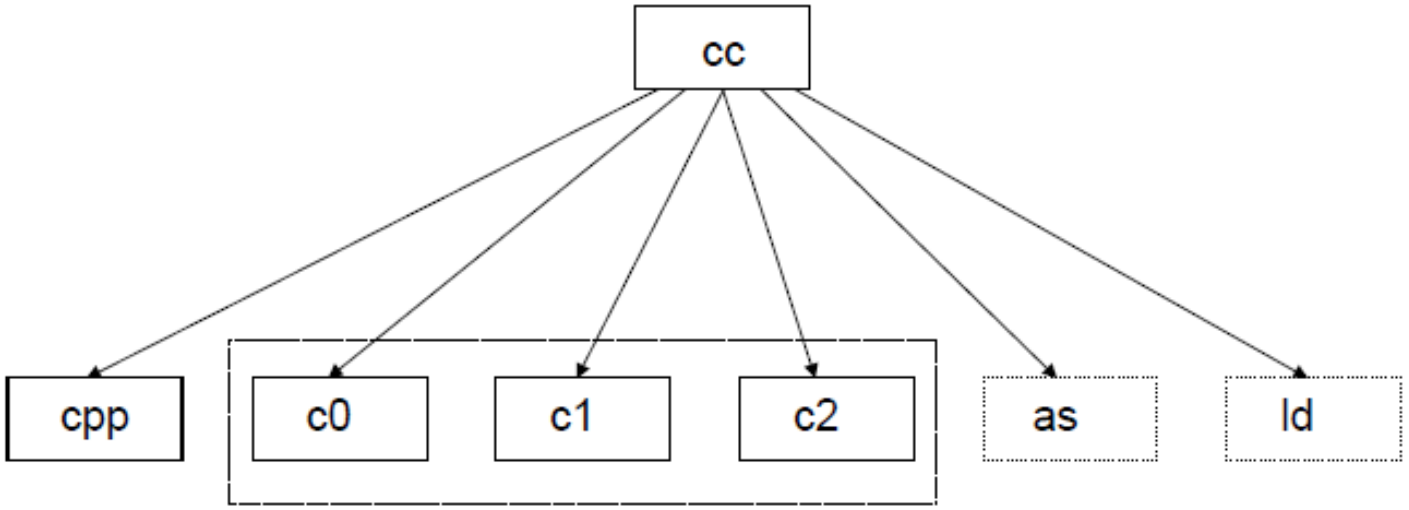


A Minimalist's Retargetable C Compiler:

PCC: Portable C Compiler

- Μεταφερτός ANSI C μεταγλωττιστής `http://pcc.ludd.ltu.se/`
- Αναβίωση του μεταγλωττιστή PQCC ο οποίος ήταν μεταγλωττιστής συστήματος στο BSD Unix
- Ο PQCC αναπτύχθηκε από τον Stephen C. Johnson, βασιζόμενος στη μεταπτυχιακή εργασία του Alan Snyder
- Πρόσφατα (από το 2007 και έπειτα) ο PCC αναβιώθηκε (νέος κώδικας και νέες στοχευόμενες αρχιτεκτονικές) από τον Anders Magnusson
- Επιλέχθηκε ως μεταγλωττιστής συστήματος για το λειτουργικό σύστημα OpenBSD
- Σχετικά μικρή βάση κώδικα
- Υποστηριζόμενες αρχιτεκτονικές
  - AMD64, ARM, HP PARISC, i386, M16C, MIPS, Nova, PDP-10, PDP-11, PowerPC, SPARCv9, Super-H, VAX

Η οργάνωση του μεταγλωττιστή PCC



cpp – The C PreProcessor

c0 – Parser and tree builder

c1 – Code generator

c2 – peephole optimizer

as – assembler

ld – linkage loader

Anders Magnusson
<ragge@ltu.se>