

Why Johnny and Jenny Can't Program in C (But We Could)

Dr. Eric Freudenthal

Brian A. Carter

October 2, 2008

Overview

- ▶ What happened to CS students' ability to program in C?
- ▶ Our approach to addressing the problem
- ▶ Teaching tricks we've picked up along the way
- ▶ The \$20 all-in-one embedded target and the free tools



Why We Could Program in C

- ▶ **Principal CS teaching language**
 - ▶ Pascal (or some other procedural language)
 - ▶ Featured explicit memory management (i.e., pointers)
 - ▶ Forced explicit management of compiler and linker
- ▶ **Connections to machine organization were made clear**
 - ▶ Memory semantics were similar
 - ▶ Those of us who knew FORTRAN had it even easier (goto statements)
- ▶ **Understanding C requires understanding of:**
 - ▶ Separate compilation (global symbols, types having to match, etc.)
 - ▶ Memory management



Java as a Principal Teaching Language

- ▶ This is not a rant against Java—instead, we’re proposing a response
- ▶ Java’s memory model is very different from machine organization
 - ▶ Global variables are “hidden” within classes
 - ▶ Automatic garbage collection
 - ▶ Objects have methods
- ▶ Tools hide “systems” issues
 - ▶ IDEs (students don’t learn/understand separate compilation)
 - ▶ Students don’t understand the roles of linker, symbol scope, etc.
- ▶ Students are “exposed” to C in a language survey course
 - ▶ But, C’s semantics are best understood in the context of architecture



Why is this a Problem?

- ▶ **Upper-division “systems” courses**
 - ▶ Students don't know how to program in C
 - ▶ Students are unfamiliar with major systems components (linker, libraries, etc.)
- ▶ **Employers are unhappy**
 - ▶ CS students don't understand runtime issues
 - ▶ EE students know C, but they can't program in the large



Our Approach

- ▶ Leave introductory sequence unchanged
 - ▶ CS I, II, and III remain Java-based
- ▶ Teach C and machine organization together (for a full semester)
 - ▶ Introduce students to the key concepts in C
 - ▶ Then use assembly/machine language to show how it's implemented
- ▶ Students seem to “get it”
 - ▶ Understand the relationship of OO memory & method abstractions to runtime
 - ▶ Students are able to learn (and even derive) compilation techniques
 - ▶ We'll show you a few pedagogical tricks that seem to work later
 - ▶ The first group of students are now attending a senior-level OS course
 - ▶ Much stronger skills are observed



Lab Course

- ▶ Students first learn UNIX tools
- ▶ Students write C programs that expose:
 - ▶ Variable storage
 - ▶ Pointers
 - ▶ Shifts & masks
- ▶ Students write leaf routines (called by C) in assembly language
- ▶ Students call C routines from assembly language
- ▶ Students learn how interrupts work

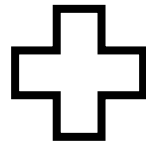


Teaching Trick 1: Reduction to goto C

- ▶ goto is legal in C
- ▶ First: perform a manual transformation
 - ▶ Students first translate for, while, etc. to goto C
 - ▶ Conversion of goto C to assembly language is trivial
- ▶ After, in an assignment, students produce standard transformation templates

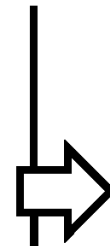
Block-structured

```
if (x==3)
    y = 4;
else
    y = 5;
```



goto C

```
if (x!=3) goto x_not_3;
y = 4;
goto end_not3;
x_not_3:
    y = 5;
end_not3:
```



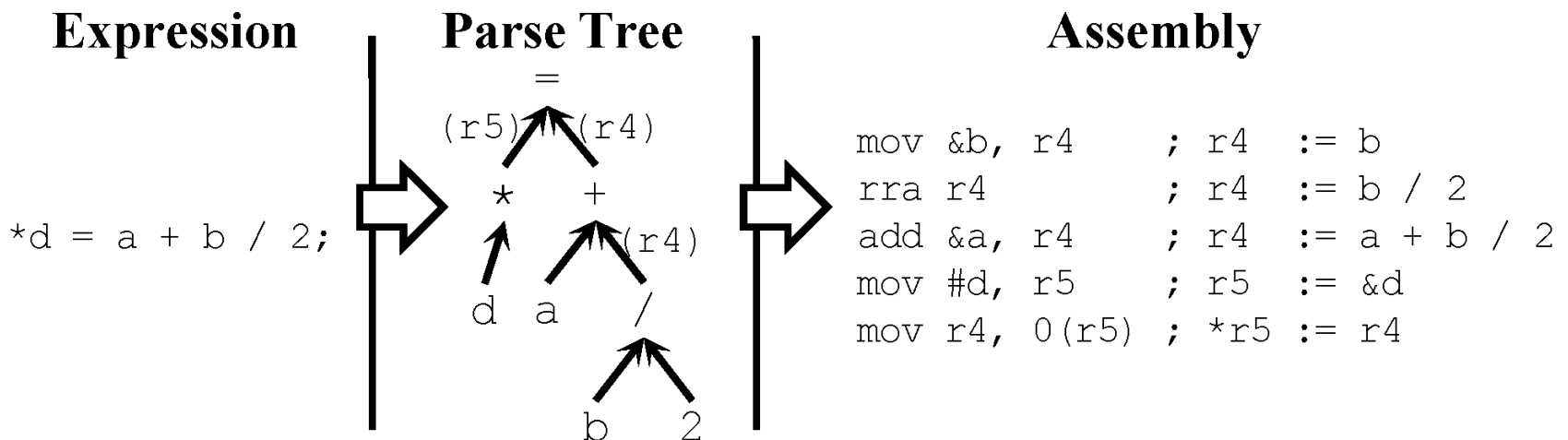
Assembly

```
cmp #3, &x
jnz x_not_3
mov #4, &y
jmp end_not3
x_not_3:
    mov #5, &y
end_not3:
```



Teaching Trick 2: Introduction of Operator (Parse) Trees

- ▶ First: challenge students to translate arithmetic expressions to assembly language
 - ▶ Have them “own the challenge”
- ▶ Next introduce operator trees



Teaching Trick 3: Two-pass Assembly

- ▶ Motivate the problem with an exercise
- ▶ Solve the problem by reserving space for operands in the first pass

C Source Code

```
top:
  if (ip->val==5) goto found_entry;
  ip = ip->next;
  goto top;
found_entry:
```

Structure Definition

```
struct Qi {
  struct Qi* next; // offset 0
  short val;       // offset 2
};
```



Corresponding Assembly Language

```
top:
  mov &ip,    r4
  cmp #5,    2(r4)
  jz  found_entry
  mov 0(r4), &ip
  jmp top
found_entry:
```

	(+5 word offset)	
	(-9 word offset)	



Embedded Target System

- ▶ MSP430 low-power embedded controller
- ▶ Twenty-seven instructions
- ▶ \$20 for full development system
- ▶ Works with completely free open-source GNU development environment

