COSC121: Computer Systems: Compiling Control Structures

Jeremy Bolton, PhD
Assistant Teaching Professor

Constructed using materials:
- Patt and Patel *Introduction to Computing Systems* (2nd)
- Patterson and Hennessy *Computer Organization and Design* (4th)

**A special thanks to Rich Squier and Robert van Engelen**
Notes

• Read PP.13 – PP.14
Outline

• Overview of
  – Compilers
    • Translation of higher level program structures to assembly
This week ... our journey takes us ...
Compiling Control Structures
Code for Conditional

Assuming all addresses are close enough that PC-relative branch can be used.
Generating Code for If-Else

- if (x)
  {
    y++;  
    z--;  
  }
- else {
    y--;  
    z++;  
  }

LDR  R0, R5, #0
BRz  ELSE  ; x is not zero
LDR  R1, R5, #-1 ; incr y
ADD  R1, R1, #1
STR  R1, R5, #-1
LDR  R1, R5, #02 ; decr z
ADD  R1, R1, #1
STR  R1, R5, #-2
JMP  DONE   ; skip else code
ELSE  LDR  R1, R5, #-1 ; decr y
ADD  R1, R1, #-1
STR  R1, R5, #-1
LDR  R1, R5, #-2 ; incr z
ADD  R1, R1, #1
STR  R1, R5, #-2
DONE   ... ; next statement
Matching Else with If

- Else is always associated with closest unassociated if.

```java
if (x != 10)
    if (y > 3)
        z = z / 2;
    else
        z = z * 2;
```

is the same as...

```java
if (x != 10) {
    if (y > 3)
        z = z / 2;
    else
        z = z * 2;
}
```

is NOT the same as...

```java
if (x != 10) {
    if (y > 3)
        z = z / 2;
}
else
    z = z * 2;
```
Chaining If’s and Else’s

• if (month == 4 || month == 6 || month == 9 || month == 11)
    
    printf(“Month has 30 days.
”);

else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12)
    
    printf(“Month has 31 days.
”);

• else if (month == 2)

• printf(“Month has 28 or 29 days.
”);

• else

• printf(“Don’t know that month.
”);
• while (test) 
  loop_body;

**While**

*Executes loop body as long as test evaluates to TRUE (non-zero).*

*Note: Test is evaluated **before** executing loop body.*
Code for Iteration

Assuming all addresses are on the same page.
Generating Code for While

\[
x = 0;
\text{while} \ (x < 10) \ \{ \\
\quad \text{printf("%d ", x);} \\
\quad x = x + 1;
\}
\]

AND \ R0, R0, #0
STR \ R0, R5, #0 ; x = 0
; test
LOOP
LDR R0, R5, #0 ; load x
ADD R0, R0, #-10
BRzp DONE
; loop body
LDR R0, R5, #0 ; load x
...
<printf>
...
ADD R0, R0, #1 ; incr x
STR R0, R5, #0
JMP LOOP ; test again
DONE ; next statement
Infinite Loops

• The following loop will never terminate:

  • \( x = 0; \)
    \( \text{while } (x < 10) \)
    \( \quad \text{printf("%d ", x);} \)

  • Loop body does not change condition, so test never fails.

  • This is a common programming error that can be difficult to find.
For

- \textbf{for (init; end-test; re-init) statement}

\textbf{Executes loop body as long as test evaluates to TRUE (non-zero). Initialization and re-initialization code included in loop statement.}

\textbf{Note: Test is evaluated \textbf{before} executing loop body.}
Generating Code for For

\[
\text{for (i = 0; i < 10; i++)} \\
\text{printf("\%d ", i);} \\
\]

; init
AND R0, R0, #0
STR R0, R5, #0 ; i = 0

; test
LOOP
LDR R0, R5, #0 ; load i
ADD R0, R0, #-10
BRzp DONE

; loop body
LDR R0, R5, #0 ; load i

...<printf>
...

; re-init
ADD R0, R0, #1 ; incr i
STR R0, R5, #0
JMP LOOP ; test again

DONE ; next statement

This is the same as the while example!
Example For Loops

1. /* -- what is the output of this loop? -- */
   for (i = 0; i <= 10; i ++)
       printf("%d ", i);

2. /* -- what does this one output? -- */
   letter = 'a';
   for (c = 0; c < 26; c++)
       printf("%c ", letter+c);

3. /* -- what does this loop do? -- */
   numberOfOnes = 0;
   for (bitNum = 0; bitNum < 16; bitNum++) {
       if (inputValue & (1 << bitNum))
           numberOfOnes++;
   }
Nested Loops

• Loop body can (of course) be another loop.

• /* print a multiplication table */
  • for (mp1 = 0; mp1 < 10; mp1++) {
      for (mp2 = 0; mp2 < 10; mp2++) {
        printf("%d\t", mp1*mp2);
      }
      printf("\n");
  }
Do-While

- do
  loop_body;
- while (test);

Executes loop body as long as test evaluates to TRUE (non-zero).

Note: Test is evaluated after executing loop body.
Problem Solving in C

• Stepwise Refinement
  – as covered in Chapter 6
• ...but can stop refining at a higher level of abstraction.

• Same basic constructs
  – Sequential -- C statements
  – Conditional -- if-else, switch
  – Iterative -- while, for, do-while
Example: Finding Prime Numbers

- Print all prime numbers less than 100.
  - A number is prime if its only divisors are 1 and itself.
  - All non-prime numbers less than 100 will have a divisor between 2 and 10.
Primes: 1st refinement

Start

Initialize

Print primes

Stop

Initialize
num = 2

num < 100

T

Print num if prime

num = num + 1

F
Primes: 2nd refinement

Initialize num = 2

num < 100

Print num if prime

num = num + 1

Divide num by 2 through 10

no divisors?

Print num
Primes: 3rd refinement

Initialize divisor = 2

F

divisor <= 10

T

Clear flag if num%divisor > 0

divisor = divisor + 1

no divisors?

T

Print num

F

Divide num by 2 through 10