Machine-Level Programming: Loops

CMPU 224 – Computer Organization
Jason Waterman
Looping in C

• There are three looping constructs in C
  • while, do-while, and for loops

• Turn loops into goto construct to implement

```c
while(test) {
  body;
}
```

```c
do {
  body;
} while(test);
```

```c
for (init; test; update; {
  body;
})
```
Looping: “do while” example

- Count number of 1’s in argument \( x \)
  - Known as the parity of the number
- Use conditional branch to either continue looping or to exit loop

Goto Version

```c
long pcount_do(unsigned long x) {
    long result = 0;
    do {
        result += x & 0x1;
        x >>= 1;
    } while (x);
    return result;
}
```

C Goto Version

```c
long pcount_goto(unsigned long x) {
    long result = 0;
    loop:
        result += x & 0x1;
        x >>= 1;
        if(x) goto loop;
    return result;
}
```
## Jumps

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Synonym</th>
<th>Jump condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jmp Label</code></td>
<td></td>
<td><code>1</code></td>
<td>Direct jump</td>
</tr>
<tr>
<td><code>je Label</code></td>
<td><code>jz</code></td>
<td><code>ZF</code></td>
<td>Equal / zero</td>
</tr>
<tr>
<td><code>jne Label</code></td>
<td><code>jnz</code></td>
<td><code>~ZF</code></td>
<td>Not Equal / not zero</td>
</tr>
<tr>
<td><code>js Label</code></td>
<td></td>
<td><code>SF</code></td>
<td>Negative</td>
</tr>
<tr>
<td><code>jns Label</code></td>
<td></td>
<td><code>~SF</code></td>
<td>Nonnegative</td>
</tr>
<tr>
<td><code>jg Label</code></td>
<td><code>jne</code></td>
<td><code>(SF ^ OF) &amp; ~ZF</code></td>
<td>Greater (signed &gt; )</td>
</tr>
<tr>
<td><code>jge Label</code></td>
<td><code>jnl</code></td>
<td><code>(SF ^ OF)</code></td>
<td>Greater or equal (signed &gt;=)</td>
</tr>
<tr>
<td><code>jl Label</code></td>
<td><code>jnge</code></td>
<td><code>SF ^ OF</code></td>
<td>Less (signed &lt;)</td>
</tr>
<tr>
<td><code>jle Label</code></td>
<td><code>jng</code></td>
<td>`(SF ^ OF)</td>
<td>ZF`</td>
</tr>
<tr>
<td><code>ja Label</code></td>
<td><code>jnbe</code></td>
<td><code>~CF &amp; ~ZF</code></td>
<td>Above (unsigned &gt;)</td>
</tr>
<tr>
<td><code>jae Label</code></td>
<td><code>jnb</code></td>
<td><code>~CF</code></td>
<td>Above or equal (unsigned &gt;=)</td>
</tr>
<tr>
<td><code>jb Label</code></td>
<td><code>jnae</code></td>
<td><code>CF</code></td>
<td>Below (unsigned &lt;)</td>
</tr>
<tr>
<td><code>jbe Label</code></td>
<td><code>jna</code></td>
<td>`CF</td>
<td>ZF`</td>
</tr>
</tbody>
</table>
“Do-While” Loop Compilation

Goto Version

```c
long pcount_goto(unsigned long x)
{
    long result = 0;
    loop:
        result += x & 0x1;
        x >>= 1;
        if(x) goto loop;
    return result;
}
```

```
movl $0, %eax # result = 0
loop:
    movq %rdi, %rdx
    andl $1, %edx # t = x & 0x1
    addq %rdx, %rax # result += t
    shrq $1, %rdi # x >>= 1
    jne loop # if (x) goto loop
    ret
```

<table>
<thead>
<tr>
<th>Register</th>
<th>Use(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>%rdi</td>
<td>Argument x</td>
</tr>
<tr>
<td>%rax</td>
<td>result</td>
</tr>
</tbody>
</table>
General “Do-While” Translation

C Code

```
do
  Body
while (Test);
```

Goto Version

```
loop:
  Body
  if (Test)
     goto loop
```

• Body:

```{Statement_1;
  Statement_2;
  ...
  Statement_n;}
```
General “While” Translation #1

• “Jump-to-test” translation
• A “do-while” loop with a jump to the first test
• Used with –Og

```
goto test;
loop:
    Body
test:
    if (Test)
        goto loop;
done:
```
While Loop Example #1

C Code

```c
long pcount_while(unsigned long x) {
  long result = 0;
  while (x) {
    result += x & 0x1;
    x >>= 1;
  }
  return result;
}
```

Jump to Test Version

```c
long pcount_goto_jtt(unsigned long x) {
  long result = 0;
  goto test;
loop:
  result += x & 0x1;
  x >>= 1;
  test:
  if(x) goto loop;
  return result;
}
```

- Compare to do-while version of function
- Initial goto starts loop at test
General “While” Translation #2

While version

while (Test)
  Body

Do-While Version

if (!Test)
  goto done;
do
  Body
  while (Test);
done:

Goto Version

if (!Test)
  goto done;
loop:
  Body
  if (Test)
    goto loop;
done:
While Loop Example #2

C Code

```c
long pcount_while(unsigned long x) {
    long result = 0;
    while (x) {
        result += x & 0x1;
        x >>= 1;
    }
    return result;
}
```

Do-While Version

```c
long pcount_goto_dw(unsigned long x) {
    long result = 0;
    if (!x) goto done;
    loop:
    result += x & 0x1;
    x >>= 1;
    if(x) goto loop;
    done:
    return result;
}
```

• Initial conditional guards the entrance to the loop
While Loop Summary

• Convert to a do-while loop that can be implemented with goto statements

• Jump-to-test
  • Jump directly to the test of the do-while loop

• Do-while conversion
  • Guard the entrance to the do-while loop with a !test
  • If the while condition is not true (!test) jump to the end of the do-while loop
“For” Loop Form

**General Form**

```
for (Init; Test; Update)

Body
```

```c
#define WSIZE 8*sizeof(long)
long pcount_for(unsigned long x){
    size_t i;
    unsigned bit;
    long result = 0;
    for (i = 0; i < WSIZE; i++){
        bit = (x >> i) & 0x1;
        result += bit;
    }
    return result;
}
```
"For" Loop $\Rightarrow$ While Loop

For Version

```plaintext
for (Init; Test; Update )

Body
```

While Version

```plaintext
Init;

while (Test ) {

Body

Update;

}
```
For-While Conversion

Init

\[ i = 0 \]

Test

\[ i < \text{FSIZE} \]

Update

\[ i++ \]

Body

\[
\{ \\
\quad \text{bit} = (x >> i) & 0x1; \\
\quad \text{result} += \text{bit}; \\
\}
\]

```
long pcount_for_while(unsigned long x){
    size_t i;
    unsigned bit
    long result = 0;
    i = 0;
    while (i < FSIZE){
        bit = (x >> i) & 0x1;
        result += bit;
        i++;
    }
    return result;
}
```
“For” Loop Do-While Conversion

**C Code**

```c
long pcount_for(unsigned long x) {
    size_t i;
    unsigned bit;
    long result = 0;
    for (i = 0; i < WSIZE; i++)
    {
        bit = (x >> i) & 0x1;
        result += bit;
    }
    return result;
}
```

- Initial test can be optimized away

**Goto Version**

```c
long pcount_for_goto_dw(unsigned long x) {
    size_t i;
    unsigned bit;
    long result = 0;
    i = 0;  // Init
    if (!(i < WSIZE))
        goto done;
    loop:
    {
        bit = (x >> i) & 0x1;
        result += bit;
    }
    i++;
    if (i < WSIZE)  // Test
        goto loop;
    done:
    return result;
}
```
For Loop Summary

• Convert to a while loop of the following form:
  Init;
  While(test)
    Body;
    Update;

• Convert while loop to a do-while

• Use goto statements to implement the do-while loop
Summary

• C Control
  • if-then-else
  • do-while
  • while, for
  • switch

• Assembler Control
  • Conditional jump
  • Conditional move
  • Indirect jump (via jump tables)
  • Compiler generates code sequence to implement more complex control

• Standard Techniques
  • Loops converted to do-while or jump-to-test form
  • Large switch statements use jump tables
  • Sparse switch statements may use decision trees (if-elseif-elseif-else)