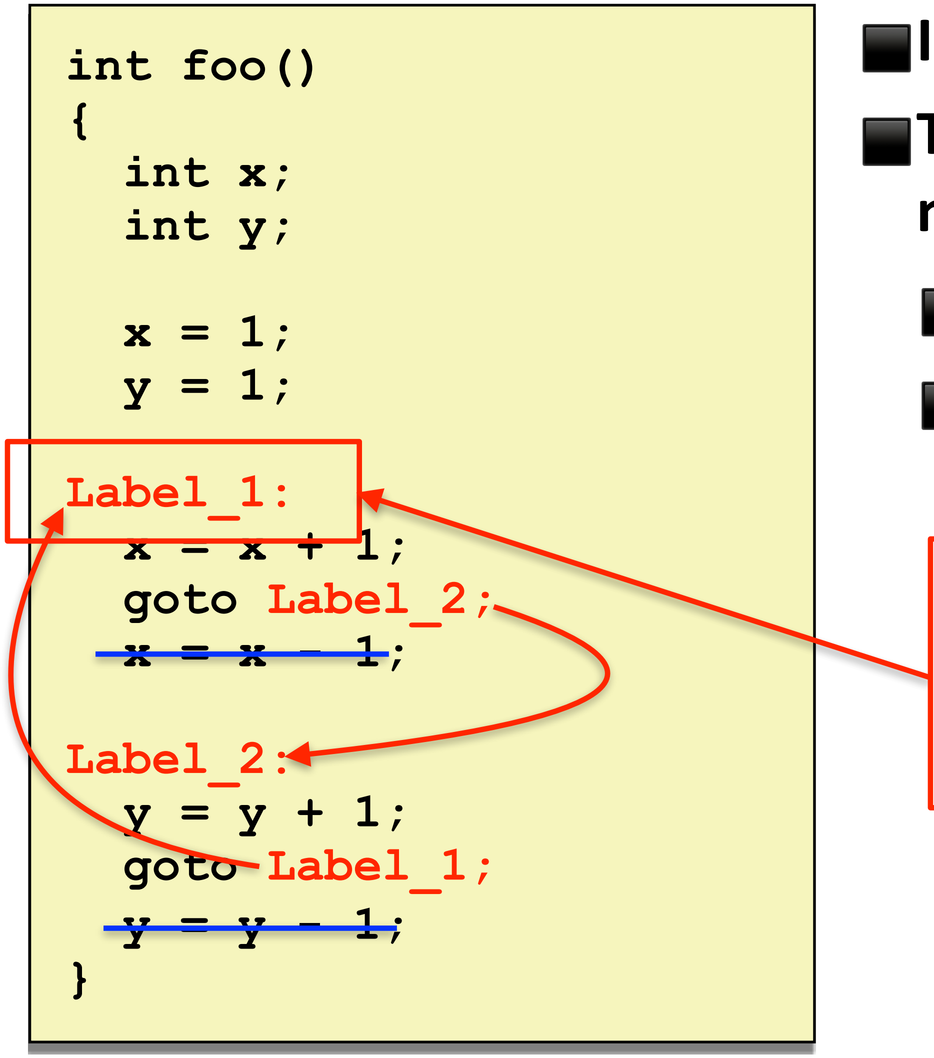
Machine-Level Programming II: Control

Today

- Control: Condition codes
- Conditional branches
- **Loops**
- Switch Statements

A Simple GoTo Example



```
int foo()
{
   int x;
   int y;

   x = 1;
   y = 1;

Label_1:
   x = x + 1;
   goto Label_2;
   x = x - 1;

Label_2:
   y = y + 1;
   goto Label_1;
   y = y - 1;
}
```

- Infinite loop
- The following two instructions are never executed
  - x = x - 1
  - y = y - 1

- Indicates a location in the code (line number)
- Not a statement

“Do-While” Loop Example

C Code

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

What is the goto version?

- **Count number of 1’s in argument `x` (“popcount”)**

"Do-While" Loop Example

C Code

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

Goto Version

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

- Count number of 1's in argument **x** ("popcount")
- Use conditional branch to either continue looping or to exit loop

“Do-While” Loop Compilation

Goto Version

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

Register	Use(s)
%rdi	Argument **x**
%rax	**result**

```
    movl    $0, %eax     #  result = 0
.L2:                     # loop:
    movq    %rdi, %rdx
    andl    $1, %edx     #  t = x & 0x1
    addq    %rdx, %rax   #  result += t
    shrq    %rdi         #  x >>= 1
    jne     .L2          #  if (x) goto loop
    rep; ret
```

General “Do-While” Translation

C Code

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

Goto Version

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

- **Body:**

```
{
   Statement1;
   Statement2;
     …
   Statementn;
}
```

Practice

C Code

```
long inc()
{
 long x = 0;
  do {
     x += 1;
   } while (x < 100);
   return x;
}
```

Goto Version

```
long inc_goto()
{
 long x = 0;
 loop:
   x += 1;
   if(x < 100) goto loop;
   return x;
}
```

How Translate "While" Loops?

While version

```
while (Test)
    Body
```

Any ideas?

General "While" Translation #1

- "Jump-to-middle" translation
- Used with –Og

While version

```
while (Test)
  Body
```

Goto Version

```
  goto test;
loop:
  Body
test:
  if (Test)
    goto loop;
done:
```

While Loop Example #1

C Code

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

Jump to Middle

```
long pcount_goto_jtm
  (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" conversion
- Used with –O1

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

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

Do-While Version

```
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;
}
```

- Compare to do-while version of function
- Initial conditional guards entrance to loop

"For" Loop Form

General Form

```
for (Init; Test; Update )

    Body
```

```
#define WSIZE 8*sizeof(int)
long pcount_for
  (unsigned long x)
{
  size_t i;
  long result = 0;
  for (i = 0; i < WSIZE; i++)
  {
    unsigned bit =
      (x >> i) & 0x1;
    result += bit;
  }
  return result;
}
```

Init

```
i = 0
```

Test

```
i < WSIZE
```

Update

```
i++
```

Body

```
{
  unsigned bit =
    (x >> i) & 0x1;
  result += bit;
}
```

"For" Loop → While Loop

For Version

```
for (Init; Test; Update)

    Body
```

How to express a for loop as a while loop?

"For" Loop → While Loop

For Version

```
for (Init; Test; Update )

    Body
```

While Version

```
Init;

while (Test ) {

    Body

    Update;

}
```

For-While Conversion

Init

```
i = 0
```

Test

```
i < WSIZE
```

Update

```
i++
```

Body

```
{
  unsigned bit =
     (x >> i) & 0x1;
  result += bit;
}
```

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

Practice

- Write the goto version

For-loop Version

```
int reduce(int *A, int size)
{
  int i;
  int result = 0;


  for (i = 0; i < size; i++)
  {
      result += A[i];
  }

  return result;
}
```

Practice

- Write the goto version

For-loop Version

```
int reduce(int *A, int size)
{
  int i;
  int result = 0;


  for (i = 0; i < size; i++)
  {
      result += A[i];
  }

  return result;
}
```

While Version

```
int reduce(int *A, int size)
{
  int i;
  int result = 0;
  i = 0;

  while (i < size)
  {
      result += A[i];
      i++;
  }

  return result;
}
```

Practice

- Write the goto version

While Version

```
int reduce(int *A, int size)
{
  int i;
  int result = 0;
  i = 0;

  while (i < size)
  {
      result += A[i];
      i++;
  }

  return result;
}
```

Do-While Version

```
int reduce(int *A, int size)
{
  int i;
  int result = 0;
  i = 0;
  if !(i < size) goto done;
  do
  {
      result += A[i];
      i++;
  } while (i < size)
done:
  return result;
}
```

Practice

- Write the goto version

Do-While Version

```
int reduce(int *A, int size)
{
   int i;
   int result = 0;
   i = 0;
   if !(i < size) goto done;
   do
   {
         result += A[i];
         i++;
   } while (i < size)
done:
   return result;
}
```

GoTo Version

```
int reduce(int *A, int size)
{
   int i;
   int result = 0;
   i = 0;
   if !(i < size) goto done;
Loop:
   {
         result += A[i];
         i++;
   } if (i < size) goto Loop;
done:
   return result;
}
```