

goto in C

The goto statement allows transfer of control to any labelled point with a function. For example, this code:

```
for (int i = 1; i <= 10; i++) {  
    printf("%d\n", i);  
}
```

can be written as:

```
int i = 1;  
loop:  
    if (i > 10) goto end;  
    i++;  
    printf("%d", i);  
    printf("\n");  
    goto loop;  
end:
```

goto in C

- goto statements can result in very difficult to read programs.
- goto statements can also result in slower programs.
- In general, use of goto is considered poor programming style.
- Do not use goto without very good reason.
- kernel & embedded programmers sometimes use goto.

MIPS Programming

Writing correct assembler directly is hard.

Recommended strategy:

- develop the solution in C
- map to “simplified” C
- translate each simplified C statement to MIPS instructions

Simplified C

- does *not* have while, compound if, complex expressions
- *does* have simple if, goto, one-operator expressions

Simplified C makes extensive use of

- *labels* ... symbolic name for C statement
- *goto* ... transfer control to labelled statement

Example:

Mapping C into MIPS

Things to do:

- allocate variables to registers/memory
- place literals in data segment
- transform C program to:
 - break expression evaluation into steps
 - replace control structures by goto

add: C to simplified C

Standard C

```
int main(void) {  
    int x = 17;  
    int y = 25;  
    printf("%d\n", x + y);  
}
```

Simplified C

```
int main(void) {  
    int x, y, z;  
    x = 17;  
    y = 25;  
    z = x + y;  
    printf("%d", z);  
    printf("\n");  
}
```

add: simplified C to MIPS

Simplified C

```
int main(void) {  
    int x, y, z;  
    x = 17;  
    y = 25;  
    z = x + y;  
    printf("%d", z);  
    printf("\n");  
}
```

MIPS

```
main:  
    li    $t0, 17  
    li    $t1, 25  
    add   $t2, $t1, $t0  
  
    move  $a0, $t2  
    li    $v0, 1  
    syscall  
  
    li    $a0, '\n'  
    li    $v0, 11  
    syscall  
  
    jr    $ra
```

while: C to simplified C

Standard C

```
i = 0;  
n = 0;  
while (i < 5) {  
    n = n + i;  
    i++;  
}
```

Simplified C

```
i = 0;  
n = 0;  
loop:  
    if (i >= 5) goto end;  
    n = n + i;  
    i++;  
    goto loop;  
end:
```

while: simplified C to MIPS

Simplified C

```
i = 0;  
n = 0;  
loop:  
    if (i >= 5) goto end;  
    n = n + i;  
    i++;  
    goto loop;  
end:
```

MIPS

```
li $t0, 0    # i in $t0  
li $t1, 0    # n in $t1  
loop:  
    bge $t0, 5, end  
    add $t1, $t1, $t0  
    add $t0, $t0, 1  
    goto loop  
end:
```

if: C to simplified C

Standard C

```
if (i < 0) {  
    n = n - i;  
} else {  
    n = n + i;  
}
```

Simplified C

```
if (i >= 0) goto else1;  
    n = n - i;  
    goto end1;  
else1:  
    n = n + i;  
end1:
```

Note: you can't use else as a label in C

if: simplified C to MIPS

Simplified C

```
if (i >= 0) goto else1;  
    n = n - i;  
    goto end1;  
else1:  
    n = n + i;  
end1:
```

MIPS

```
# assume i in $t0  
# assume n in $t1  
bge $t0, 0, else1  
sub $t1, $t1, $t0  
goto end1  
else1:  
add $t1, $t1, $t0  
end1:
```

if/and: C to simplified C

Standard C

```
if (i < 0 && n >= 42) {  
    n = n - i;  
} else {  
    n = n + i;  
}
```

Simplified C

```
if (i >= 0) goto else1;  
if (n < 42) goto else1;  
    n = n - i;  
    goto end1;  
else1:  
    n = n + i;  
end1:
```

if/and: simplified C to MIPS

Simplified C

```
if (i >= 0) goto else1;  
if (n < 42) goto else1;  
    n = n - i;  
    goto end1;  
else1:  
    n = n + i;  
end1:
```

MIPS

```
# assume i in $t0  
# assume n in $t1  
bge $t0, 0, else1  
blt $t1, 42, else1  
sub $t1, $t1, $t0  
goto end1  
else1:  
add $t1, $t1, $t0  
end1:
```

odd-even: C to simplified C

Standard C

```
if (i < 0 || n >= 42) {
    n = n - i;
} else {
    n = n + i;
}
```

Simplified C

```
if (i < 0) goto then1;
if (n >= 42) goto then1;
goto else1;
then1:
    n = n - i;
    goto end1;
else1:
    n = n + i;
end1:
```

Example Printing First 10 Integers

```
int main(void) {
    for (int i = 0; i <= 10; i++) {
        printf("%d\n", i);
    }
}
```

Example Printing First 10 Integers

Convert to goto and simple C statements and decide where variables will be stored.

```
int main(void) {
    int i;    // in register $t0
    i = 0;

loop:
    if (i >= 10)
        goto end;
    i++;
    printf("%d", i);
    printf("%c", '\n');
    goto loop;
end:
    return 0;
}
```

Example Printing First 10 Integers

```
main:                                # int main(void) {
                                     # int i; // in register $t0
    li    $t0, 0                    # i = 0;
loop:                                # loop:
    bge   $t0, 10 end               # if (i >= 10) goto end;
    add   $t0, $t0, 1               # i++;
    move  $a0, $t0                  # printf("%d" i);
    li    $v0, 1
    syscall
    li    $a0, '\n'                # printf("%c", '\n');
    li    $v0, 11
    syscall
    b     loop                      # goto loop;
end:
    jr    $ra                      # return
```