Control Structures (Practice)

ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

(q1) If-then-Else in Assembly

- Translate to assembly

```
signed int eax, ebx;
. . .
if (eax < ebx) {
  eax = ebx;
} else {
  ebx = 12;
}
```

(q1) Solutions

- Translate to assembly

```
signed int eax, ebx;
. . .
if (eax < ebx) {
  eax = ebx;
} else {
  ebx = 12;
}
```

```
  cmp eax, ebx
  jge else
  mov eax, ebx
  jmp endif
else:
  mov ebx, 12
endif:
```

- Note that the simpler version with possible one extra mov instruction doesn't work here
  - Because the registers that are updated are also used in the conditional

(q2) If-then-Else in Assembly

- Translate to assembly

```
unsigned int eax;
unsigned short ebx;
. . .
if (eax < 100) {
  ebx = eax;
} else {
  ebx = 12;
}
```

(q2) Solutions

- Translate to assembly

```
unsigned int eax;
unsigned short ebx;
. . .
if (eax < 100) {
  ebx = eax;
} else {
  ebx = 12;
}
```

```
  cmp eax, 100
  jae else
  mov ebx, eax
  jmp endif
else:
  mov ebx, 12
endif:
```

(q3) If-then-Else Reverse Engineering

- Translate to C (use register names as variable names, assume all signed)

```
    cmp eax, -1
    jnz label_1
    mov ebx, eax
    jmp end
label_1:
    mov ebx, -12
end:
```

(q3) Solutions

- Translate to C (use register names as variable names, assume all signed)

```
    cmp eax, -1
    jnz label_1
    mov ebx, eax
    jmp end
label_1:
    mov ebx, -12
end:
```

```
if (eax == -1) {
  ebx = eax;
} else {
  ebx = -12;
}
```

(q4) If-then-else++

- Translate to assembly

```
if (((edx >= 2) && (ecx > 1)) || (ebx == 3)) {
    eax = 10;
} else {
    eax = 20;
}
```

(q4) Solution

```
if (((edx >= 2) && (ecx > 1)) || (ebx == 3)) {
    eax = 10;
} else {
    eax = 20;
}
```

```
cmp ebx, 3
je  then
cmp edx, 2
jl  else
cmp ecx, 1
jle else
then:
  mov eax, 10
  jmp end
else:
  mov eax, 20
end:
```

- Note that the short-circuiting is the other way around than the example in the lecture notes (which used a top-level && rather than a top-level ||)

(q5) Do-while-loop in Assembly

- Translate to assembly

```
signed int eax;
. . .
do {
  eax += 2;
} while (eax > 0)
```

(q5) Solution

- Translate to assembly

```
signed int eax;
. . .
do {
  eax += 2;
} while (eax > 0)
```

```
begin:
  add eax, 2
  cmp eax, 0
  jg begin
```

(q6) For-loop in Assembly

- Translate to assembly

```
signed int eax, ebx;
. . .
for (eax = 10; eax < 100; eax++, ebx--) {
  eax += ebx
}
```

(q6) Solutions

- Translate to assembly

```
signed int eax, ebx;
. . .
for (eax = 10; eax < 100; eax++, ebx--) {
  eax += ebx
}
```

```
  mov eax, 10
begin:
  cmp eax, 100
  jge endfor
  add eax, ebx
  inc eax
  dec ebx
 jmp begin
endfor:
```

(q7) For-loop Reverse Engineering

- Translate to C (use register names as variable names, assume all signed)

```
mov eax, 100
begin:
  cmp ebx, eax
  jz nope
  add ebx, 1
nope:
  sub eax, 10
  cmp eax, 7
  jge begin
```

(q7) Solution

- Translate to C (use register names as variable names, assume all signed)

```
mov eax, 100
begin:
   cmp ebx, eax
   jz nope
   ebx += 1
nope:
   sub eax, 10
   cmp eax, 7
   jge begin
```

```
for (eax = 100; eax >= 7;
      eax -= 10) {
   if (ebx != eax) {
     ebx += 1;
   }
}
```

- This translation is correct because both the assembly code and the C code do at least one iteration. But if we were to change the number 100 to say, the number 2, on both codes, then they would differ: the assembly code would do the if, but the C code wouldn't! So a do-while loop is technically more correct…

(q7) Solution

- Translate to C (use register names as variable names, assume all signed)

```
mov eax, 100
begin:
   cmp ebx, eax
   jz nope
   ebx += 1
nope:
   sub eax, 10
   cmp eax, 7
   jge begin
```

```
eax = 100;
do {
   if (ebx != eax) {
      ebx++;
   }
   eax -=10;
} while (eax >= 7)
```

- Do-while loop above is now equivalent to the assembly code, even if we change the numbers to different values. It's also a more straightforward translation.