COMP 122/L Lecture 10

Mahdi Ebrahimi

Slides adapted from Dr. Kyle Dewey

Outline

- Translating complex `if` statements
  - Complex conditions
- Translating `while` loops

Translating complex `if` statements

Branch (`b`) Instruction

Branch (`b`) can be used to jump to code with a label.

Code can be given labels, just as with data.

Branch (`b`) Instruction

Branch (`b`) can be used to jump to code with a label.

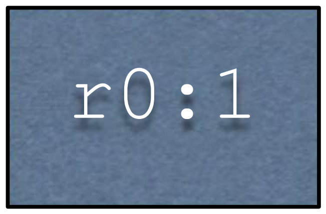
Code can be given labels, just as with data.

```
  mov r0, #1
  b other
  mov r0, #5
other:
  mov r1, r0
```

Branch (b) Instruction

Branch (b) can be used to jump to code with a label.

Code can be given labels, just as with data.



```
→  mov r0, #1
   b other
   mov r0, #5
other:
   mov r1, r0
```

–Start at first instruction, as usual

Branch (b) Instruction

Branch (b) can be used to jump to code with a label.

Code can be given labels, just as with data.

r0:1

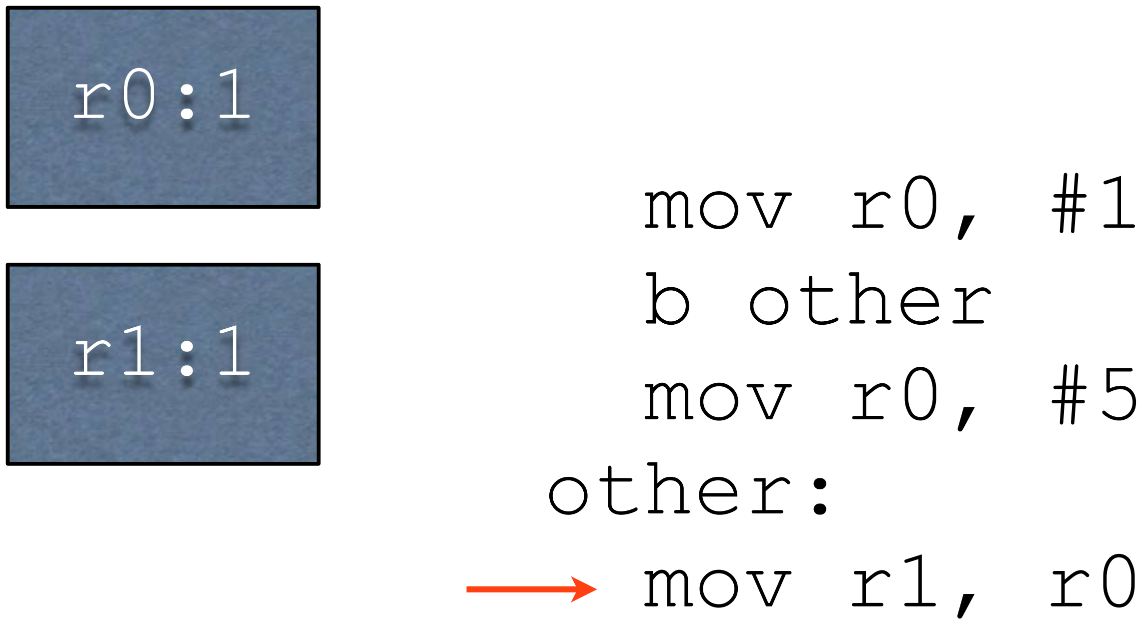
```
      mov r0, #1
  →   b other
      mov r0, #5
    other:
      mov r1, r0
```

–Execution of b other causes execution to jump to other

Branch (b) Instruction

Branch (b) can be used to jump to code with a label.

Code can be given labels, just as with data.



```
    mov r0, #1
    b other
    mov r0, #5
other:
    mov r1, r0
```

-Execution of b other causes execution to jump to other
-The mov r0, #5 instruction is never touched

Translating `if`

Key point: the `b` instruction can be conditionally executed.

Translating `if`

Key point: the `b` instruction can be conditionally executed.

```
  mov r0, #0
  mov r1, #5
  cmp r1, #5
  beq elsewhere
  mov r0, #25
elsewhere:
  mov r2, r0
```

Translating `if`

Key point: the `b` instruction can be conditionally executed.

```
r0:0
```

```
→      mov r0, #0
       mov r1, #5
       cmp r1, #5
       beq elsewhere
       mov r0, #25
   elsewhere:
       mov r2, r0
```

Translating `if`

Key point: the `b` instruction can be conditionally executed.

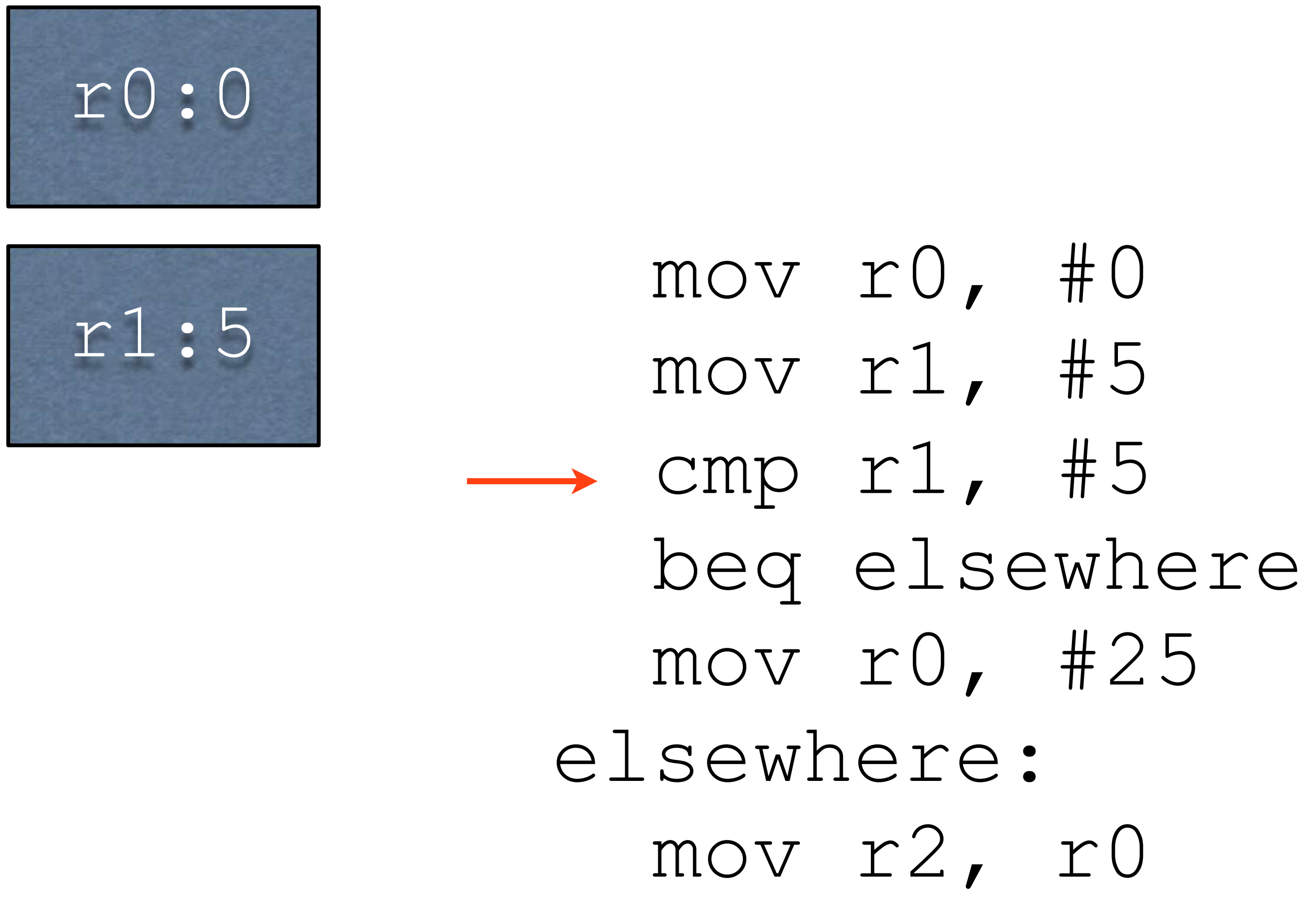
r0:0

r1:5

```
   mov r0, #0
   mov r1, #5
   cmp r1, #5
   beq elsewhere
   mov r0, #25
elsewhere:
   mov r2, r0
```

Translating `if`

Key point: the `b` instruction can be conditionally executed.



```
    mov r0, #0
    mov r1, #5
    cmp r1, #5
    beq elsewhere
    mov r0, #25
elsewhere:
    mov r2, r0
```

–Does 5 – 5, sets the zero bit

Translating `if`

Key point: the `b` instruction can be conditionally executed.

r0:0

r1:5

```
      mov r0, #0
      mov r1, #5
      cmp r1, #5
  →   beq elsewhere
      mov r0, #25
    elsewhere:
      mov r2, r0
```

–Because the zero bit is set, the jump occurs

Translating `if`

Key point: the `b` instruction can be conditionally executed.

```
r0:0

r1:5

r2:0
```

```
     mov r0, #0
     mov r1, #5
     cmp r1, #5
     beq elsewhere
     mov r0, #25
elsewhere:
→    mov r2, r0
```

–Because the zero bit is set, the jump occurs
–The mov r0, #25 instruction is never executed

Utility for `if`

- More convenient to translate long `if`s with labeled branches
- Basically required for nested `if` or complex conditions
- Conditionally-executed instructions are most useful for short `if`s
    - Arguably the common case

Example:

`absolute_value_label.s`

Nested `if`

- Can be handled with multiple comparisons and branches
- Tricky part: assembly is written in a linear way, but branches are inherently non-linear
- Example:
  - `NestedIf.java`
  - `nested_if.s`

Complex Conditions

Boolean Operations

Boolean operations (e.g., `&&`, `||`) require multiple checks.

Boolean Operations

Boolean operations (e.g., `&&`, `||`) require multiplechecks.

```
if (x == 0 || x == 5) {
  y = 0;
} else if (min <= x && x <= max) {
  y = 1;
}
```

```
if (x == 0 || x == 5) {
  y = 0;
} else if (min <= x && x <= max) {
  y = 1;
}
```

```
    if(x == 0 || x == 5) {
       y = 0;
    } else if (min <= x && x <= max) {
      y = 1;
    }
```

```
if (x == 0) {
  y = 0;
} else {
  if (x == 5) {
    y = 0;
  } else {
    if (min <= x) {
      if (x <= max) {
        y = 1;
      }}}}
```

Example:

`BigIf.java`

`big_if.s`

Translating `while` Loops

Translating `while`

- Lot like `if`, but with jumps to the start/end
- Example:
  - `WhileLoop.java`
  - `while_loop.s`