

# COMP 122/L Lecture 9

Kyle Dewey

# Outline

- The compare (`cmp`) instruction
- Conditionally-executed instructions
- Translating simple `if` statements

# The compare (cmp) instruction

# Compare (cmp)

Subtracts two given operands, discarding the result. However, the status bits (e.g., carry, zero, etc.) get set.

# Compare (cmp)

Subtracts two given operands, discarding the result. However, the status bits (e.g., carry, zero, etc.) get set.

---

```
mov  r0, #5  
cmp  r0, #5
```

# Compare (cmp)

Subtracts two given operands, discarding the result. However, the status bits (e.g., carry, zero, etc.) get set.

---

```
mov r0, #5
```

```
cmp r0, #5
```

Sets zero bit/flag  
(result is zero)

# Compare (cmp)

Subtracts two given operands, discarding the result. However, the status bits (e.g., carry, zero, etc.) get set.

---

```
mov r0, #5
```

```
cmp r0, #5
```

**Sets zero bit/flag  
(result is zero)**

---

```
mov r0, #5
```

```
cmp r0, #20
```

# Compare (cmp)

Subtracts two given operands, discarding the result. However, the status bits (e.g., carry, zero, etc.) get set.

---

```
mov r0, #5
```

```
cmp r0, #5
```

**Sets zero bit/flag**  
**(result is zero)**

---

```
mov r0, #5
```

```
cmp r0, #20
```

**Sets negative bit/flag**  
**(result is negative)**



# Significance

Status bits say something about  
the result of arithmetic comparisons

# Significance

Status bits say something about  
the result of arithmetic comparisons

---

```
mov r0, #5
```

```
cmp r0, #5
```

**Sets zero bit/flag**  
**(result is zero)**

# Significance

Status bits say something about  
the result of arithmetic comparisons

---

```
mov r0, #5
```

```
cmp r0, #5
```

Sets zero bit/flag  
(result is zero)

**Operands must have been equal.**

# Significance

Status bits say something about  
the result of arithmetic comparisons

---

```
mov r0, #5
```

```
cmp r0, #5
```

Sets zero bit/flag  
(result is zero)

**Operands must have been equal.**

---

```
mov r0, #5
```

```
cmp r0, #20
```

Sets negative bit/flag  
(result is negative)

# Significance

Status bits say something about  
the result of arithmetic comparisons

---

```
mov r0, #5
```

```
cmp r0, #5
```

Sets zero bit/flag  
(result is zero)

**Operands must have been equal.**

---

```
mov r0, #5
```

```
cmp r0, #20
```

Sets negative bit/flag  
(result is negative)

**First operand must be < second.**

# Conditionally-executed instructions

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
movmi r0, #42
```



# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
movmi r0, #42
```

move if the negative bit is set

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
movmi r0, #42
```

move if the negative bit is set

---

```
movp1 r1, #23
```

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

`movmi r0, #42`

move if the negative bit is set

---

`movpl r1, #23`

move if the negative bit is **not** set

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
moveq r0, #42
```

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
moveq r0, #42
```

move if the zero bit is set

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

```
moveq r0, #42
```

move if the zero bit is set

---

```
movne r0, #42
```

# Conditionally-Executed Instructions

ARM allows for instructions to be *conditionally* executed, depending on the values of the status bits.

---

`moveq r0, #42`

move if the zero bit is set

---

`movne r0, #42`

move if the zero bit is **not** set

**Example:**

`conditional_execution.s`



# Translating simple `if` statements

# Translating `if`

- Simple `if`s can be translated with conditionally-executed instructions
- Example:
  - `AbsoluteValue.java`
  - `absolute_value.s`