MIPSUnit: A Unit Testing Framework for MIPS Assembly
Grand Valley State University
Zachary Kurmas & Jack Rosenhauer II

MIPSUnit Overview

- Quickly and easily verify correctness of MIPS assembly code
  - Fast grading!
  - Grading is faster, but not automatic
  - Instructor must still comment on style

- Makes it easy for students to verify the correctness of their own assembly code
  - Find and fix mistakes instead of “writing off” points
  - “Testing across the curriculum”
  - Students find testing difficult
  - There is no substitute for lots of practice
  - The more courses that require testing, the more proficient students become

MUnit

MUnit is JUnit (Java) based

- Familiar syntax for students
- Lists each failure
- Tests can’t affect each other
- Each test runs in a new “sandbox”
- Less expressive than MSpec

MSpec

MSpec is RSpec (Ruby) based

- Much more expressive than MUnit
- Easy to write exhaustive tests
- Generates assembly file containing tests
- Students must learn a little Ruby to write their own tests
- Only reports first failure
- Tests can affect each other
- No way to re-set initial memory states between tests

MUnit Usage

```java
public class DateFashionTest {
    @Test
    public void you_unstylish_date_ unstylish() {
        int[] expected = {6, 5, 4, 3, 2, 1, 0};
        int[] observed = getWords(array1, 0, 7);
        Assert.assertEquals(expected, observed);
        Assert.assertEquals(11, get(0));
    }
}
```

```java
@RunWith(SpringJUnit4ClassRunner.class)
public class DateFashionTest extends org.springframework.test.context.web.WebIntegrationTest {
    @Test
    public void you_unstylish_date_ unstylish() {
        int[] expected = {6, 5, 4, 3, 2, 1, 0};
        int[] observed = getWords(array1, 0, 7);
        Assert.assertEquals(expected, observed);
        Assert.assertEquals(11, get(0));
    }
}
```

MSpec Usage

```ruby
describe date_fashion do
  let(:you) { 2 }
  let(:date) { 10 }
  context "when you is equal to or less than date" do
    it "returns 0 if you unstylish and date is ok" do
      expect(date_fashion(you, date)).to eq(0)
    end
  end
  context "when you is less than date and greater than 2" do
    it "returns 2 if you and your date are stylish" do
      expect(date_fashion(you, date)).to eq(2)
    end
  end
  context "when you is greater than date and you are unstylish" do
    it "returns 100 if you is greater than date and you are unstylish" do
      expect(date_fashion(you, date)).to eq(100)
    end
  end
end
```

Contact Information

Zachary Kurmas
kurmasz@gvsu.edu
http://www.cis.gvsu.edu/~kurmasz/Software/

Jack Rosenhauer II
jackrosenhauer@gmail.com

MUnit Command Line

- Compile tests, then run
  - $ javac -cp munit.jar InRangeTest.java
  - $ java -jar munit.jar In_range_error.asm InRangeTest.class

- Failure: verify true at min(InRangeTest): expected:<1> but was:<0>
- Tests run: 5, Failures: 1
- Uses familiar JUnit output format