MIPS

Prologue and Epilogue

For complete detail see Handout8 of cs654

Summary of calling sequence

```
# evaluate receiver this into 0($fp)
    sw   $a0 0($fp)
# foreach actual in actuals
    # <Evaluate actual>
    sw $a0, 0($sp)            # store argument
    addiu $sp, $sp, -4        # decrement stack pointer
# load receiver this into $a0
    lw $a0, 0($fp)
# jump AND link to method label
    jal method_label
    # return address or address of the next instruction is in $ra now
#   ...
# <Prologue>
#   ...
```

Prologue

Remember `$fp` holds the previous value of `$sp`

```
# make space for n + 3 words (3 because of $fp, $ra and $s0)
#   save frame pointer
#   save this reference
#   save return address
# update frame pointer
    addiu $fp, $sp, 4
    move $s0 $a0
# <Method Body>
```

Epilogue

After the method body is done, the return value is in `$a0`.

```
#   restore frame pointer
#   restore this reference
#   restore return address
# increment $sp by `n + m + 3`
# jump back the instruction right after jal
        jr      $ra
```

Example `Main.cool`

Lets look at an actual Cool program compiled into MIPS by reference cool compiler

```
class Main() {

  {
    fib(3)
  };

  def fib(n: Int): Int = if (n == 0) 1 else n * fib(n - 1);
}
```

Code `Main.s`

`Main.Main` or Main constructor

```
    move          $a0 $s0         # line 5
    sw   $a0 0($fp)
    la   $a0 int_lit1   # line 4
    sw   $a0 0($sp)
    addiu         $sp $sp -4
    lw   $a0 0($fp)     # line 5
    jal  Main.fib
```

Code `Main.cool` Cont.

Method Main.fib Prologue

```
Main.fib:
    addiu         $sp $sp -24
    sw    $fp 24($sp)
    sw    $s0 20($sp)
    sw    $ra 16($sp)
    addiu         $fp $sp 4
    move          $s0 $a0
```

Code `Main.cool` Cont.

Method Main.fib Epilogue

```
    lw   $fp 24($sp)
    lw   $s0 20($sp)
    lw   $ra 16($sp)
    addiu          $sp $sp 28
    jr   $ra
```