EECS483 D10: Project 4 Overview

Chun-Hung Hsiao

March 22, 2013

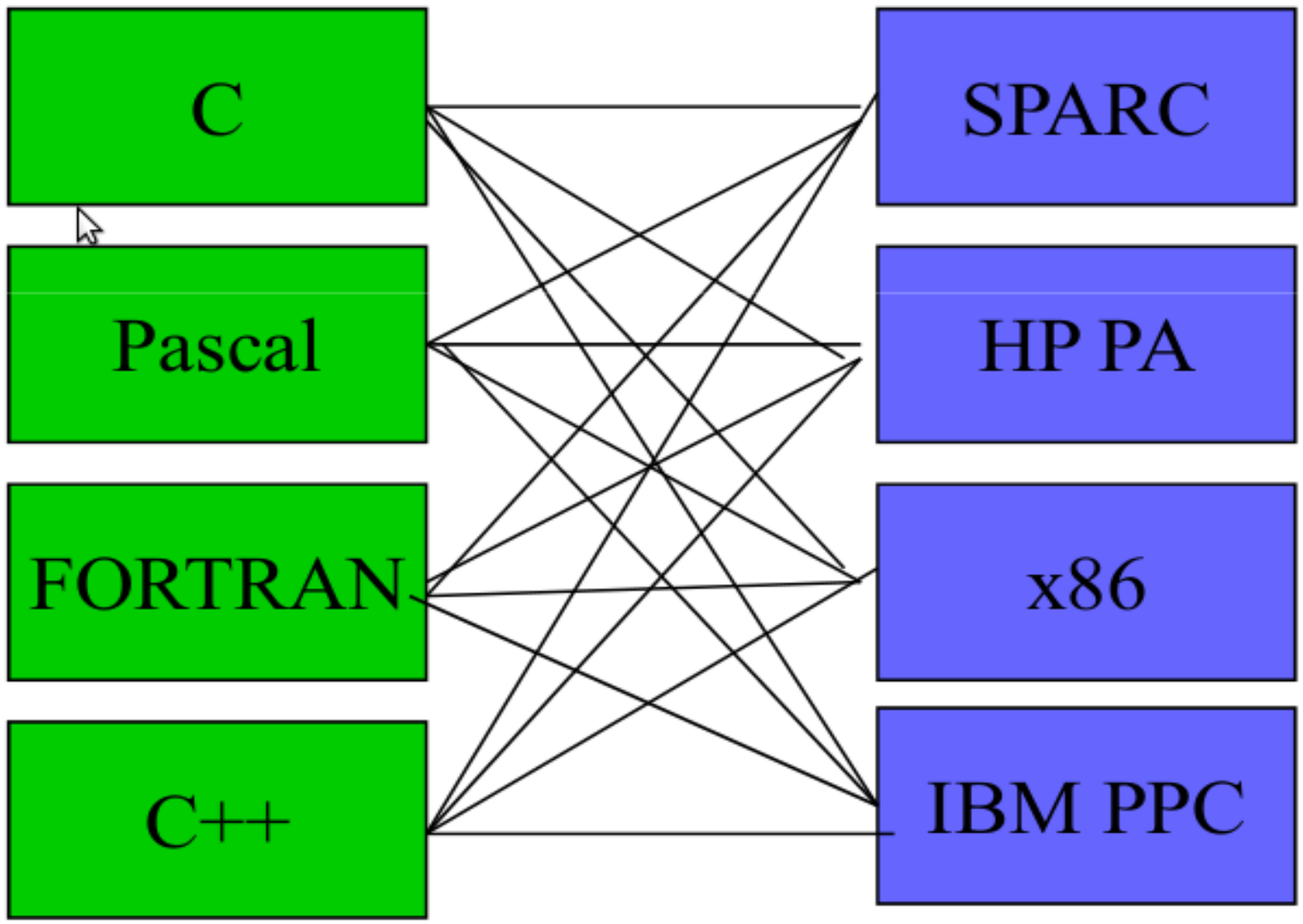
Special thanks to Ashutosh

Announcements

- Project 4 released due on 4/1
- Homework will be released soon

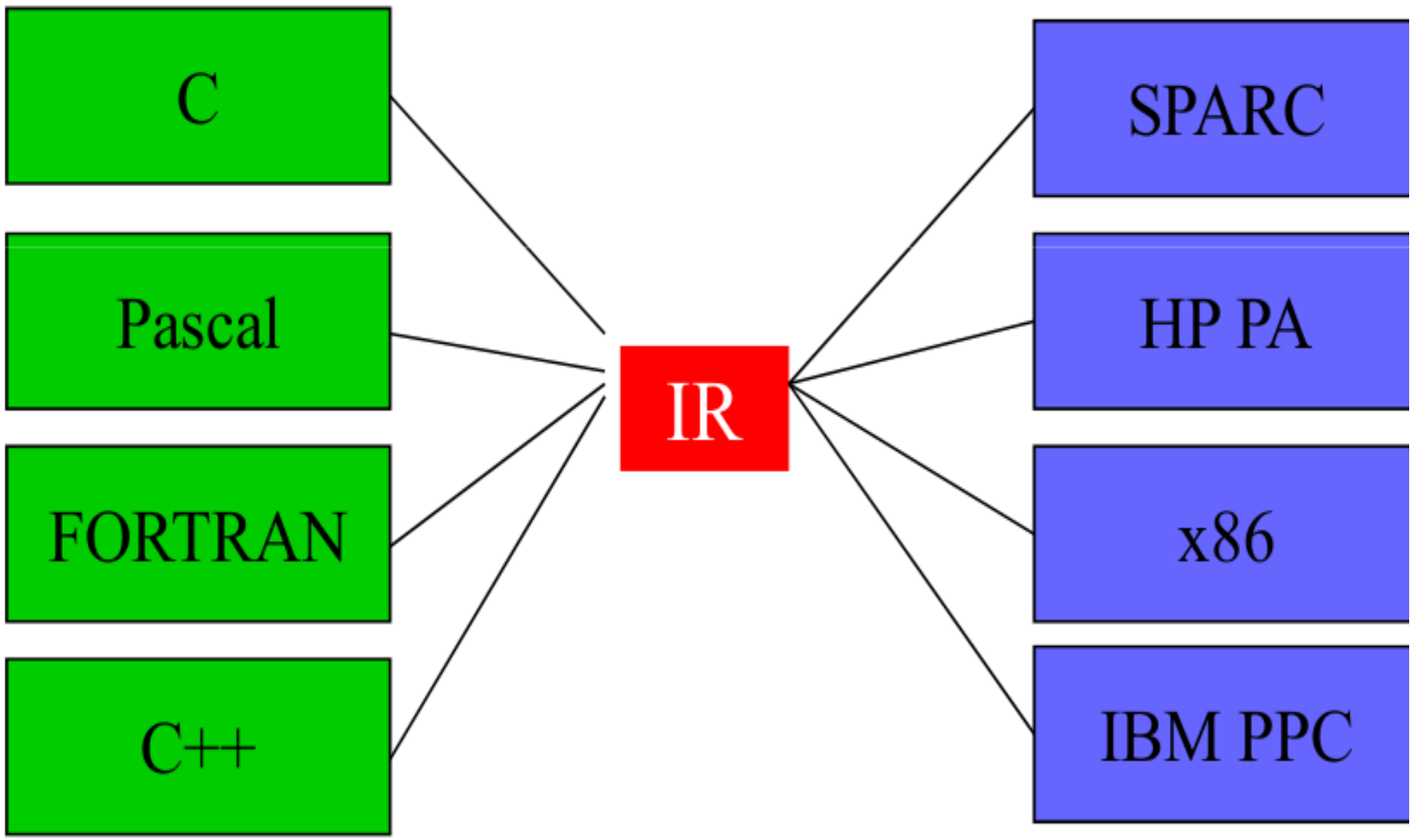
Code Generation

- Translating AST to ASM



Code Generation

- Translating AST to IR to ASM



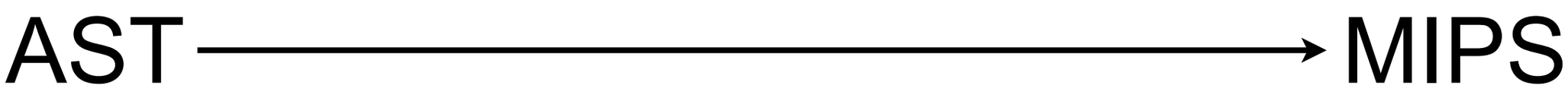
Intermediate Representation

- Close enough to most assembly languages
- General enough to hide machine-dependent details
- Easy to do post-IR analysis and optimization
  - Control flow analysis
  - Data flow analysis
  - Register allocation
  - Instruction scheduling
  - Dead code/common expression elimination
  - Code motion for speculative execution
  - And more

3-Address Code

- Close to many assembly languages
  - Easy to translate to 2-address code for x86
- General to hide machine-dependent details
  - Use temporary variables (virtual registers) instead of real registers
  - Provide a general instruction set

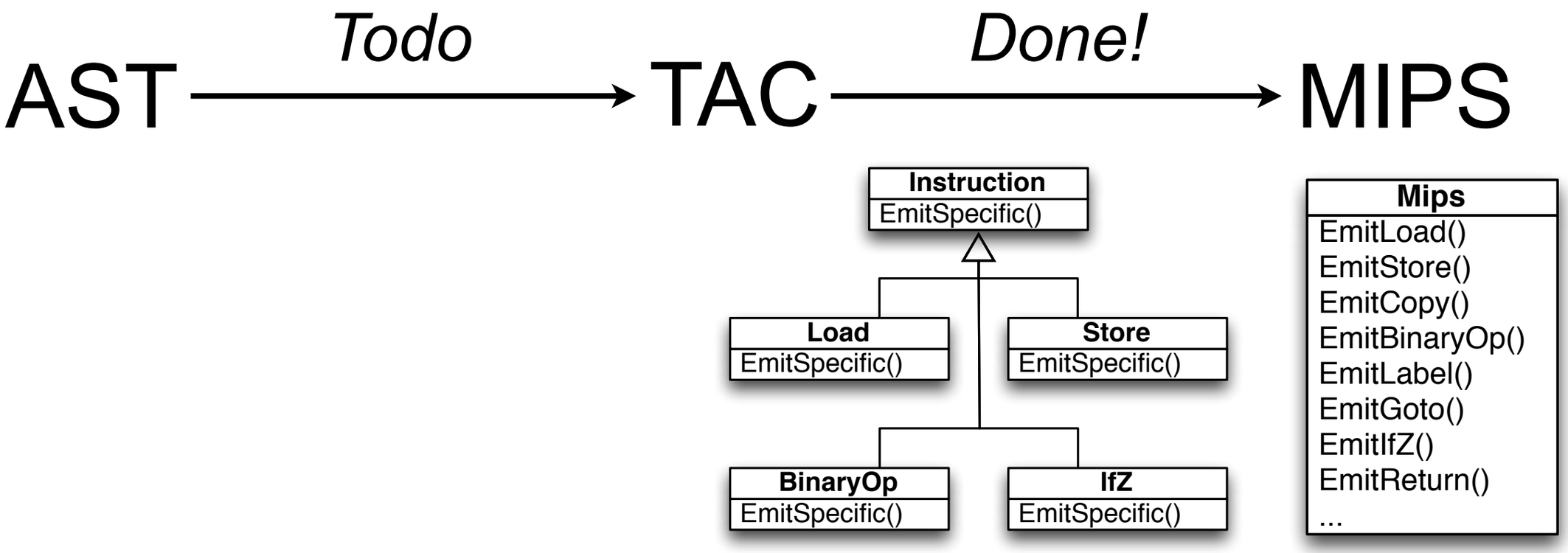
Project 4 Overview



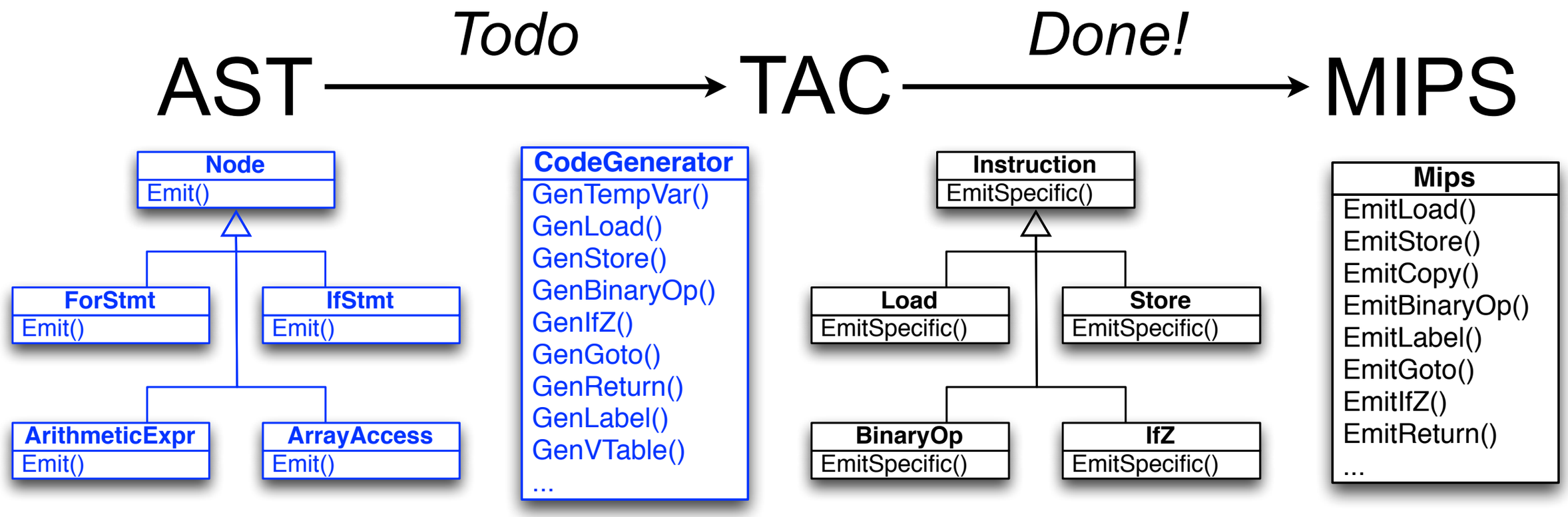
Project 4 Overview

Project 4 Overview

Project 4 Overview



Project 4 Overview



Good News

- All variables, parameters, and pointers are 32-bit
  - No double
  - Use 4 bytes to store a bool
- No interfaces!
  - Just support single inheritance to simplify vtable generation
  - You can try to implement interfaces as long as your MIPS code passes our test!

TAC Example: Control Structure

```
if (a < b + c)
    a = a - c;
c = b * c;
```

```
    _t0 = b + c;
    _t1 = a < _t1;
    IfZ _t1 Goto _L0;
    _t2 = a - c;
    a = _t2
_L0:
    _t3 = b * c;
    c = _t3;
```

TAC Example: Function Call

```
int foo(int a, int b){
    return a + b;
}

void main(){
    int c;
    int d;
    foo(c, d);
}
```

```
_foo:
    BeginFunc 4;
    _t0 = a + b;
    Return _t0;
    EndFunc;

main:
    BeginFunc 8;
    PushParam d;
    PushParam c;
    LCall _foo;
    PopParams 8;
    EndFunc;
```

TAC Example: Classes

```
class Animal{
    int height;
    void Init(int h){
        height = h;
    }
    void Speak(){}
}

class Cat extends Animal{
    void Speak(){
        Print("Meow");
    }
}
```

```
.data

_Animal.vtable:
    .word _Animal.Init
    .word _Animal.Speak

_Cat.vtable:
    .word _Animal.Init
    .word _Cat.Speak

_str0:
    .asciiz "Meow"
```

```
.text

_Animal.Init:
    BeginFunc 0;
    Store this + 4, h
    EndFunc;

_Animal.Speak:
    BeginFunc 0;
    EndFunc;

_Cat.Speak:
    BeginFunc 0;
    PushParam _str0;
    LCall _Print;
    PopParams 4;
    EndFunc;
```

TAC Example: Dynamic Dispatch

```
class Animal{
    int height;
    void Init(int h){
        height = h;
    }
    void Speak(){}
}

class Cat extends Animal{
    void Speak(){
        Print("Meow");
    }
}

void Touch(Animal a){
    a.Speak();
}
```

```
.data

_Animal.vtable:
    .word _Animal.Init
    .word _Animal.Speak

_Cat.vtable:
    .word _Animal.Init
    .word _Cat.Speak

.text

_Touch:
    BeginFunc 8;
    _t0 = Load a + 0;
    _t1 = Load _t0 + 4;
    PushParam a;
    ACall _t1;
    PopParams 4;
    EndFunc;
```

TAC Example: New Object

```
class Animal{
    int height;
    void Init(int h){
        height = h;
    }
    void Speak(){}
}

class Cat extends Animal{
    void Speak(){
        Print("Meow");
    }
}

void main(){
    Animal a;
    a = New(Cat);
}
```

```
.data

_Animal.vtable:
    .word _Animal.Init
    .word _Animal.Speak

_Cat.vtable:
    .word _Animal.Init
    .word _Cat.Speak

.text

main:
    BeginFunc 16;
    _t0 = LoadConstant 8;
    PushParam _t0;
    _t1 = LCall _Alloc;
    PopParams 4;
    _t2 = LoadLabel _Cat.vtable;
    Store _t1 + 0, _t2;
    EndFunc;
```

From TAC to MIPS

```
void main(){
    int a;
    int b;
    int c;
    if (a < b + c)
        a = a - c;
    c = b * c;
}
```

```
main:
    BeginFunc 28;
    _t0 = b + c;
    _t1 = a < _t1;
    IfZ _t1 Goto _L0;
    _t2 = a - c;
    a = _t2
_L0:
    _t3 = b * c;
    c = _t3;
    EndFunc;
```

```
        .text
        .align 2
        .globl main
main:
    # BeginFunc 28
      subu $sp, $sp, 8
      sw $fp, 8($sp)
      sw $ra, 4($sp)
      addiu $fp, $sp, 8
      subu $sp, $sp, 28
    # _tmp0 = b + c
      lw $t0, -12($fp)
      lw $t1, -16($fp)
      add $t2, $t0, $t1
      sw $t2, -20($fp)
    # _tmp1 = a < _tmp0
      lw $t0, -8($fp)
      lw $t1, -20($fp)
      slt $t2, $t0, $t1
      sw $t2, -24($fp)
    # IfZ _tmp1 Goto _L0
      lw $t0, -24($fp)
      beqz $t0, _L0
    # _tmp2 = a - c
      lw $t0, -8($fp)
      lw $t1, -16($fp)
      sub $t2, $t0, $t1
      sw $t2, -28($fp)
    # a = _tmp2
      lw $t2, -28($fp)
      sw $t2, -8($fp)
_L0:
    # _tmp3 = b * c
      lw $t0, -12($fp)
      lw $t1, -16($fp)
      mul $t2, $t0, $t1
      sw $t2, -32($fp)
    # c = _tmp3
      lw $t2, -32($fp)
      sw $t2, -16($fp)
    # EndFunc
      move $sp, $fp
      lw $ra, -4($fp)
      lw $fp, 0($fp)
      jr $ra
```

SPIM: A MIPS32 Simulator

- Since there is no MIPS machine for you to test your program, we uses SPIM simulator instead
- http://pages.cs.wisc.edu/~larus/spim.html
- Or you can find it under ~chhsiao/Public/spim

- Usage: spim -file program.s
- Demo time!

In Addition to Assembling...

- Simple linking error checking

```
*** Linker: function 'main' not defined
```

- Two Runtime checking

```
Decaf runtime error: Array size is <= 0
Decaf runtime error: Array subscript out of
bounds
```