

Plan for Learning about Stack Frames

Typical C stack frame

- gcc calling convention described in A.5 and A.6 in appendix on assembly code and MIPS

Stack frame the MiniJava compiler will generate

- Need to match the Wisconsin C-- compiler to implement garbage collection

General stack frame concept

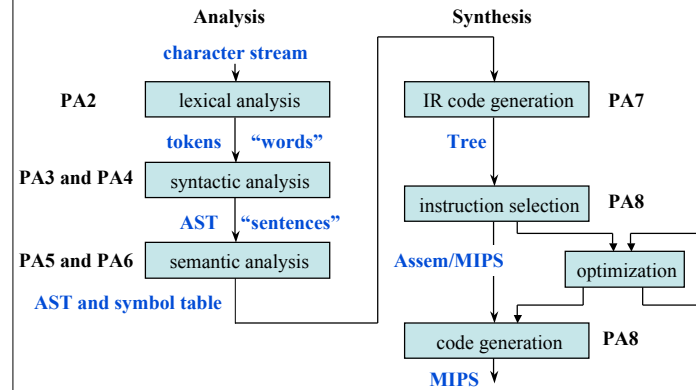
- agreement amongst programmers, procedure call convention
- what about gcc for x86?
- handling nested procedures
- handling first class functions

CS453 Lecture

C-- Stack Frame Layout

1

Structure of the MiniJava Compiler



CS453 Lecture

C-- Stack Frame Layout

2

Mapping out the stack frame for the funcCall1 example

```

int foo(int x,int y,int *z) {
    int a;
    a = x * y - *z;
    return a;
}
void main() {
    int x;
    x = 2;
    cout << foo(4,5,&x);
    cout << "\n";
}
    
```

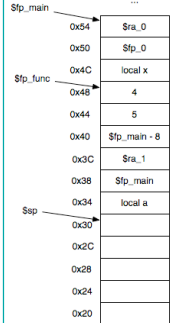
```

.text
.foo:
sw   $ra, 0($sp)    #PUSH
subu $sp, $sp, 4
sw   $fp, 0($sp)    #PUSH
subu $sp, $sp, 4
addu $fp, $sp, 4
addu $fp, $sp, 20
subu $sp, $sp, 24
...
lw   $t0, -20($fp)
move $v0, $t0
lw   $ra, -12($fp)
move $t0, $fp
lw   $fp, -16($fp)
move $sp, $t0
jr   $ra
    
```

```

.text
.globl main
main:
sw   $ra, 0($sp)    #PUSH
subu $sp, $sp, 4
sw   $fp, 0($sp)    #PUSH
subu $sp, $sp, 4
addu $fp, $sp, 8
subu $sp, $sp, 12
li   $t0, 2
sw   $t0, -8($fp)
li   $t0, 4
sw   $t0, 0($sp)    #PUSH
subu $sp, $sp, 4
li   $t0, 5
sw   $t0, 0($sp)    #PUSH
subu $sp, $sp, 4
sw   $t0, 0($sp)    #PUSH
subu $sp, $sp, 4
sw   $t0, 0($sp)    #PUSH
subu $sp, $sp, 4
jal  .foo
move $a0, $v0
...
lw   $ra, 0($fp)
move $t0, $fp
lw   $fp, -4($fp)
move $sp, $t0
jr   $ra
    
```

Stack frame for funcCall1.c



return value is put in \$v0
\$sp is set to current \$fp before return

C-- Stack Frame Layout

Wisconsin C-- calling convention

Calling convention (contract between caller and callee)

- \$sp must be divisible by 4
- caller should pass parameters in order on the stack
- upon callee entry, the stack pointer \$sp should be pointing at the first empty slot past the last parameter
- upon callee exit, the stack pointer \$sp should be pointing at the first parameter
- upon callee exit, return value should be in \$v0

Rules to follow for PA6 (to standardize frame usage)

- \$sp should always be pointing at next empty slot on the stack
- \$ra and \$fp should be stored right after the parameters on stack, you can't use any other callee-saved registers
- \$fp should be made to point at the first parameter, so that the address for the first parameter is \$fp-0, the address for the second parameter is \$fp-4, ...
- locals should be stored in order, right after \$ra and \$fp

CS453 Lecture

C-- Stack Frame Layout

4

Another example: where does each variable go?

```
class A {
public static void main(String[] a){
    System.out.println(42);
}
}

class B {
int [] x;
boolean mBool;

public int foo(boolean p1, int p2, B b, int [] y)
{
    boolean v1; int i; int j; return 0;
}
public B bar()
{
    B b;
    b = new B;
    return b;
}
public boolean baz() {
    return mBool;
}
}
```

CS453 Lecture

C-- Stack Frame Layout

5

Determining locations for vars

Local vars

- maintain counter for method that is initialized to 0
- store counter in a temporary variable
- **decrement** current counter by size of the local variable
- return the value in the temporary variable

Class members

- maintain counter for method that is initialized to 0
- store counter in a temporary variable
- **increment** current counter by size of the local variable
- return the value in the temporary variable

CS453 Lecture

C-- Stack Frame Layout

6