Handout # 8
Call Frame Structure

Assume a method has m formal parameters and requires n temporaries (local variables and partial results) to do its work. Then, after its frame is set up, the frame will have the following structure:

	⋮	⋮
`$sp`	`$fp` - 4	free space (initially)
	`$fp`	temporary 0
	⋮	⋮
	`$fp` + $4i$	temporary i
	⋮	⋮
	`$fp` + $4(n-1)$	temporary $n-1$
`$sp` + $4(n+1)$	`$fp` + $4n$	Caller's (return) address
`$sp` + $4(n+2)$	`$fp` + $4(n+1)$	Caller's self
(before `$sp`) `$sp` + $4(n+3)$	`$fp` + $4(n+2)$	Caller's frame pointer
	`$fp` + $4(n+3)$	actual parameter $m-1$
	⋮	⋮
	`$fp` + $4(n+3+m-j-1)$	actual parameter j
	⋮	⋮
(after `$sp`) `$sp` + $4(n+3+m)$	`$fp` + $4(n+3+m-1)$	actual parameter 0
	⋮	⋮

Calling Sequence

The caller performs the following actions:

1. Evaluate receiver (object whose method is to be called) and save in temp.
2. For each parameter (left to right):
   (a) evaluate,
   (b) store where `$sp` is pointing, and
   (c) decrement `$sp` by the word size (4 bytes).
3. Load the receiver object reference into `$a0`.
4. If null, set filename and line number, and jump to `dispatch_abort`
5. Jump to the method's code while leaving the return address in `$ra`. (The "jump and link" instructions accomplish this in one step.)

The callee must then perform the following actions in the method *prologue*: code generated before the real business of the method:

1. Decrement the stack pointer to accommodate $n + 3$ words where n is the number of temporaries required.
2. Save `$fp`, `$s0` (the caller's "self" register) and `$ra` in the indicated places.
3. (If the garbage collector is running, clear the temporaries: store zero to initialize each temporary.)
4. Set the frame pointer correctly.
5. Move `$a0` into the "self" register `$s0`.

After the method body is done, the return value is sitting in `$a0`. The the callee must perform the method *epilogue* which removes the call frame *including the actual parameters* from the stack:

1. Restore the caller's `$fp`, `$s0` and `$ra` from the stack.
2. Increment the stack pointer to remove $n+3+m$ words from the stack (where m is the number of parameters).
3. Jump to the location indicated in `$ra`.

Since the callee has already removed the call frame and parameters from the stack and has restored the caller's frame pointer and "self" pointer, the caller is now free to continue with its own business.