Patt & Pattel Book Stackframe for MIPS (example)

```c
int foo(int x, int y, int *z) {
    int a;
    a = x * y - *z;
    return a;
}

void main() {
    int x;
    x = 2;
    printf("%d\n", foo(4, 5, &x));
}
```

Showing the stack right before return from foo.
- $ra is the return address register
- $fp is the frame pointer register
- $sp is the stack pointer register

Patt and Patel book calling convention (for MIPS)

**Calling convention (contract between caller and callee)**
- caller should push parameters right to left onto the stack
- upon callee entry, the stack pointer $sp should be pointing at the first parameter
- upon callee exit, the stack pointer $sp should be pointing at the return value, which should be followed by the first parameter
- $sp must be divisible by 4 (for MIPS)
- $sp should always be pointing at the top entry on the stack

**Standardizing the stack frame implementation for this course**
- $ra and $fp should be stored on top of the return value slot
- locals should be stored on top of $ra and $fp
- $fp should be made to point at the first local variable, so that the address for the first local is $fp+0, the address for the second local is $fp+4, ...
- The offsets for the incoming parameters will differ based on whether there is a return value. If there is a return value, then the first parameter will be at $fp-16, the second at $fp-20, etc. If there is no return value, then the first parameter will be at $fp-12, the second at $fp-16, etc.