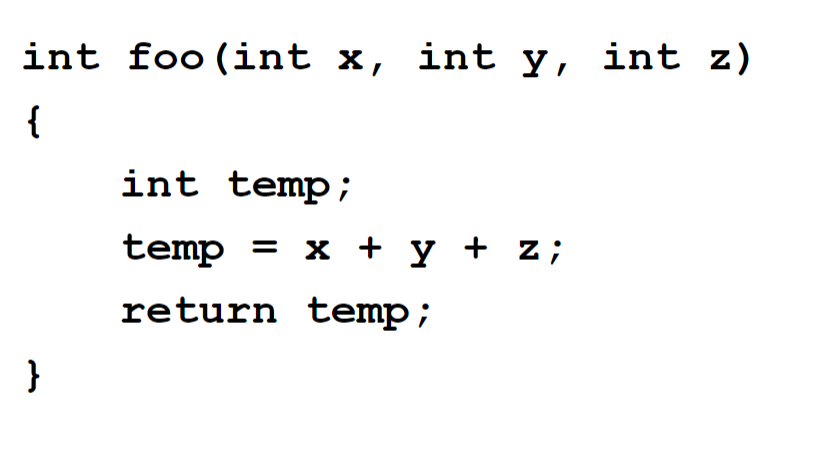
Stack Uses

* Calling subroutine
* Where do we store registers?
* Where do we store values?
* Called a “Calling Convention”
  + Different calling convention for LC-3, x86, ARM, etc.
* Templated available for if-else statements, while loops, for loops, etc.

LC-3 Calling Convention

* Assume we are making a call like
  + Y = foo(a,b,c)
* Caller: Push args onto stack right to left
* Caller: Jump to subroutine
* Callee: Move SP to leave room for Ret Val
* Callee: Move SP and store R7 (Ret Addr)
* Callee: Store any saved registers and also local variables for function…



|  |
| --- |
|  |
|  |
| R6: SP -> Local variable: temp |
| Return Address |
| Return Value |
| Copy of a |
| Copy of b |
| Copy of c |

--Stack Frame (above)

X is at SP3 (a)

Interrupt:

* Modification to the hardware of the datapath and I/O system and additional software to allow an external device to cause the CPU to stop current execution and execute a “service” routine and then resume execution of the original program
* Interrupts can be significantly more efficient than polling and are especially useful in an environment where there are numerous devices and multiple concurrent activities
* Polling is appropriate where there is a high likelihood of quick success or the CPU has nothing better to do

Recursion

Assumptions

* Stack has been initialized and R6 is pointing to the top of the stack
* The (assembly language) caller has placed arguments onto the stack right to left and then called the subroutine using a JSR or a JSRR thus the return address is already in R7
* You should also assume that R5 contains the caller's Frame Pointer and when you return back to her she will want it to contain the same value for the Frame Pointer as was there when she did the call