University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 220: Computer Systems & Programming

Call Interface Specification

Author Designs the Interface to Assembly Subroutines

Imagine that you are writing a subroutine.

How is information passed into the subroutine?

(For example, two operands to multiply, or a number to be printed.)

How is information returned from the subroutine?

(For example, the product of a multiplication, or a number read from the keyboard.)

The **answers are completely up to you** (the subroutine author).

Where Are Input/Output Bits to Subroutines Stored?

What are your options?

registers

specific memory locations

memory locations pointed to by registers or by specific memory locations

Each type can be used for input, or for output, or for both.

Examples of Assembly Subroutine Interfaces*

READNUM (on the web page)
- Read a number from the keyboard as 2's complement.
- Returns number read in R0.

PRINT_SLOT (from MP1): R1 (input) holds the slot number to be printed.

PRINT_CENTERED (also from MP1): R1 (input) holds the string to be printed.

*These interfaces are NOT up to you…sorry.

Author Designs the Interface to Assembly Subroutines

What about other registers and memory locations?

(That is, ones that your code does not use as input nor as output.)

Again, the **answers are completely up to you** (the subroutine author).

Be sure to document your choices!

Have You Attended ECE Talent Night?

I need a break.
May I **borrow** a piece of paper?
I need more paper.
May I **borrow** a textbook?
Have you ever brooded on the fact …*
…that when a friend **borrows** a ~~Klee~~
facial tissue, they never
think to return it?

*Apologies to Jack Handey.

Does Borrow Mean What I Think It Means?

Borrow: one word, many meanings!

Ambiguity is ok in English and other human languages.

We figure it out.

Ambiguity is NOT ok in digital systems.

When you write a subroutine, make ownership of registers and side effects clear!

The Caller Calls the Callee

Here are a few terms to help…

Let's say that code **A** calls subroutine **B**.

A is the **caller**.

B is the **callee**.

Each Register is Either Caller- or Callee-Saved

(Remember: code **A** calls subroutine **B**.)

If a register might be changed by **B**,
- code **A** is responsible for
- copying the bits elsewhere before calling **B**.
- Such a register is **caller-saved**.

If subroutine **B** guarantees
- that a register does not change,
- that register is **callee-saved**.
- Note: **B can still use the register**, but must save and restore the original contents to do so.

R7 and Output Registers are Always Caller-Saved

In LC-3 subroutines,
- **R7 is always caller-saved.**
- JSR(R) changes it,
- so the subroutine cannot preserve its value.

Any output register
- **is also caller-saved**,
- since the subroutine changes the register's value
- as part of completing its task.

Also Document Any Side Effects

What are side effects?

- Any change to memory (other than to memory used exclusively by the subroutine)
- Any I/O performed by the subroutine (except for I/O specified as part of the subroutine's behavior).

Always document any side effects clearly.

Subroutine Calling Interface Includes Four Parts

Together, specification of
- subroutine inputs,
- subroutine outputs,
- ownership of other registers, and
- side effects

form the **call interface** (or calling convention*) for the subroutine.

*I find it strange to have a 'convention' for one subroutine, but you may hear it used.

