CS 201 Lecture 14
Turing and Function Calls

Spring 2014
Plan for today

- So when is our instruction set “done”?
- Supporting functions in assembly
ISA design requirements

Capability
• Are some computations impossible because of missing instruction set features?

Convenience
• What operations and programming practices will users most want to use in the system? Are they easy and efficient?
ISA Capability

Relevant questions for designers and users of computers

• How do we know whether we’re leaving out something important?
• Out of all programs that could be written, which are the ones that our computer can handle?
Some of the greatest CS contributions of all time

• Alonzo Church, 1934: A particular method of describing computational algorithms ($\lambda$-calculus) was capable of describing a huge collection of important algorithms

• Alan Turing, 1936-7: A particular kind of hypothetical machine (new called a Turing Machine) could compute something iff Church’s $\lambda$-calculus could describe it
• Stephen Kleene, 1943, 1952: Any *computable algorithm* is both expressible in a basic programming language and computable by a Turing Machine.
  – A machine that can compute anything the Turing Machine could compute is called “Turing-Complete”

*Great, so what does that mean?*

• We cannot build a more capable machine than a Turing Complete machine
• If our instruction set is Turing Complete, then “capability” is not a problem
Is my ISA Turing complete?

Yes. (almost certainly)

Examples of Turing-Complete operations:

- Subtract, branch if negative.
- Increment, decrement, branch if nonzero.
- Increment, decrement.
  - if you allow for self-modifying code

Access to memory is required, but instructions can modify memory directly instead of using registers

- MIPS, x86, Java, and our Lab 6 processor are all Turing Complete
What should an ISA make convenient?

• Think about programs you have written or would like to write. What kinds of things should an instruction set make convenient?
  – Arithmetic
  – Variable accesses
  – Array accesses
  – Strings
  – Functions and code organization
  – Pointers and data structures

• This is where we start looking for ISA ideas
Function Calls in Assembly

• A function (subroutine in assembly) is a piece of code that we would like to reuse multiple times

Addresses

    //Some instructions
    X j myFunction
    X=4 //Some more instructions
    Y j myFunction
    Y+4 //Some more
    myFunction:
    ... // Some instructions
    J ?
Functions require non-static jumps

- Can’t determine where we need to go at the end of the function ahead of time
- Need to save data to describe where to “return” when the function is over

- Return address – the instruction to where a function should return
  - Depends on from where the function was called this time