October 24: CS61 Lecture 15
Lucy Cheng

Calling Convention

- How C compiler matches functions to instructions.
- Stack is part of calling convention.
- Not a law. You can imagine compiling a different way. Some don't use the stack.
- Examples:
  - %eax is the return value.
  - First argument is @ 4(%esp) when the function begins.
  - Return address is @(%esp).
  - Stack frames must be 16-bit aligned.
- Allow multiple compilers to work together.
- Callee-saved registers.
  - Registers that a normal, well-behaved function will not touch.
  - %ebp, %esp, %ebx, %esi, %edi are callee-saved register.

Structure

- %esp = top of the stack
  - May change as the function executes.
- %ebp is the base pointer register.
  - Compiler often reserves another register that is consistent.
  - Boundary between parameters and locals.
  - Doesn't change.
- f(int a, int b)
- _____
- | ret addr | a | b |
- _____
- ^ %esp
- pushl %ebp.
  - Saves base pointer at start. At the end, pop and restore value.
- pushl x = subl $4, %esp + movl x, (%esp)
- popl x = movl(%esp),x + addl $4, %esp
- _____
- ret addr g | | old ebp | ret addr | a | b |
- _____
- ^ %esp ^%ebp
- leave = movl %ebp, %esp + popl %ebp
- Last argument has highest address.

Lecture Code

l15/f42.s
- Calls g.

l15/43.s
- Clang automates.

l15/f44.s
- Calls g three times.

l15/f45.s
- Prints Hello! I love you. This message is false.
- $.L01 loads address of string into register.

l15/f46.s
- Function with two arguments. Prepare to call sum by moving arguments. Calls sum(a,b).

l15/f47.c
- clang. Smart enough to know same arguments, so just jumps to sum.

l15/f48.s
- 8 arguments to sum.

l15/f54.s
- sum function. Sums two arguments.

l15/f55.s
- int f(int* x) {return x[0] + x[1];}

l15/f56.s
- int f(int* a, int* b) {return *a + *b;}

l15/f57.s
- int f(int a[], int b, int c) {return a[b] + a[c];}

l15/f58.s
- Array of structs. Adding second elements of two structs in the array.

l15/f59.s
- leal. Loads effective address.
- Returns address of element in an array of elements of size 8.

l15/f65.s

- Loops. Returns sum from 1 to n.

l15/f66.s

- 3 arguments.
- Searches for an element. Two exits from the loop. &&. Only executes if the left is true.
- Compiler optimizes a lot, so a loop with one exit might have 2 exits in the object code.

l15/f68.s

- Sums elements of an array from 0 to n.
- Compares addresses instead of indices.

l15/f71.s

- Factorial function.

l15/f72.s

- Clang. Uses tail recursion for factorial

l15/f73.s

- While true loop. Actually a goto.
- Doesn’t actually take all the memory. Somehow does something else.

Next Unit: How to protect against attacks on OS.