

Procedures II

CSE 351 Autumn 2023

Instructor:

Justin Hsia

Teaching Assistants:

Afifah Kashif

Malak Zaki

Bhavik Soni

Naama Amiel

Cassandra Lam

Nayha Auradkar

Connie Chen

Nikolas McNamee

David Dai

Pedro Amarante

Dawit Hailu

Renee Ruan

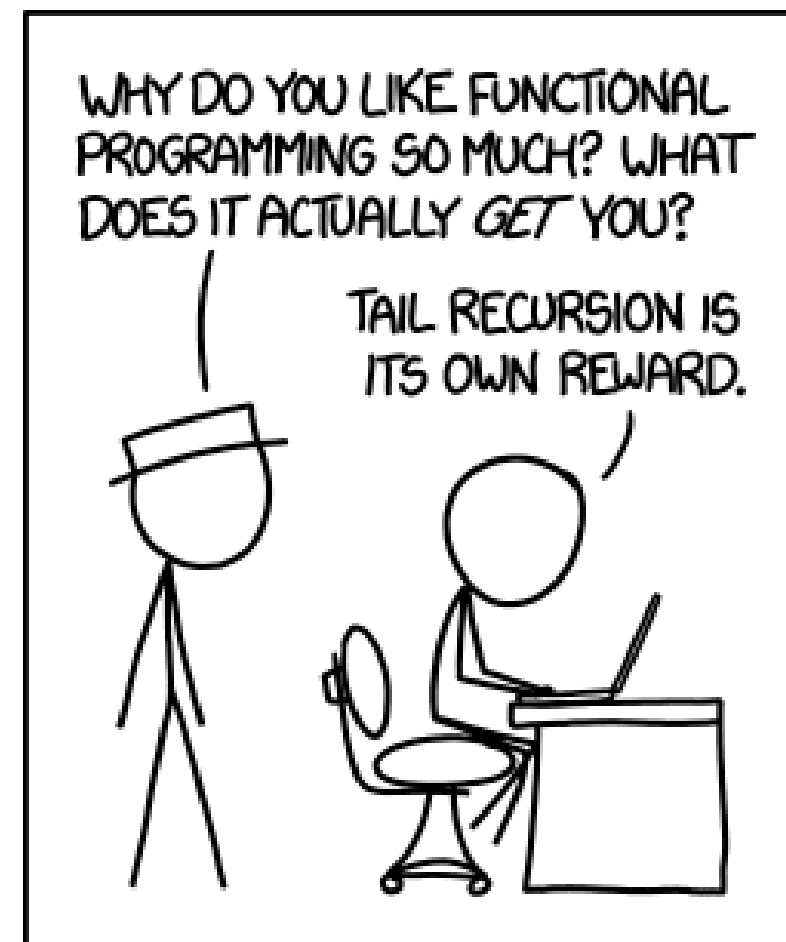
Ellis Haker

Simran Bagaria

Eyoel Gebre

Will Robertson

Joshua Tan



<http://xkcd.com/1270/>

Relevant Course Information

- ❖ Lab 1b grades released later this week
 - Regrade requests open ~24 hours after grade release (rounded to 12:00 am), close ~72 hours after grade release (rounded to 11:59 pm)
- ❖ Lab 2 due Friday (10/27)
 - Since you are submitting a text file (`defuser.txt`), there won't be any Gradescope autograder output about compilation this time – check the Code tab after submission to make sure that everything looks right
 - Extra credit (bonus) needs to be submitted to the extra credit assignment
- ❖ Midterm (take home, 11/2–11/4)
 - Make notes and use the [midterm reference sheet](#)
 - Form study groups and look at past exams!

A detailed, colorful micrograph of a microchip die, showing a complex grid of circuitry and various colored regions (purple, blue, yellow, green, red) representing different functional blocks and interconnects.

Procedures II

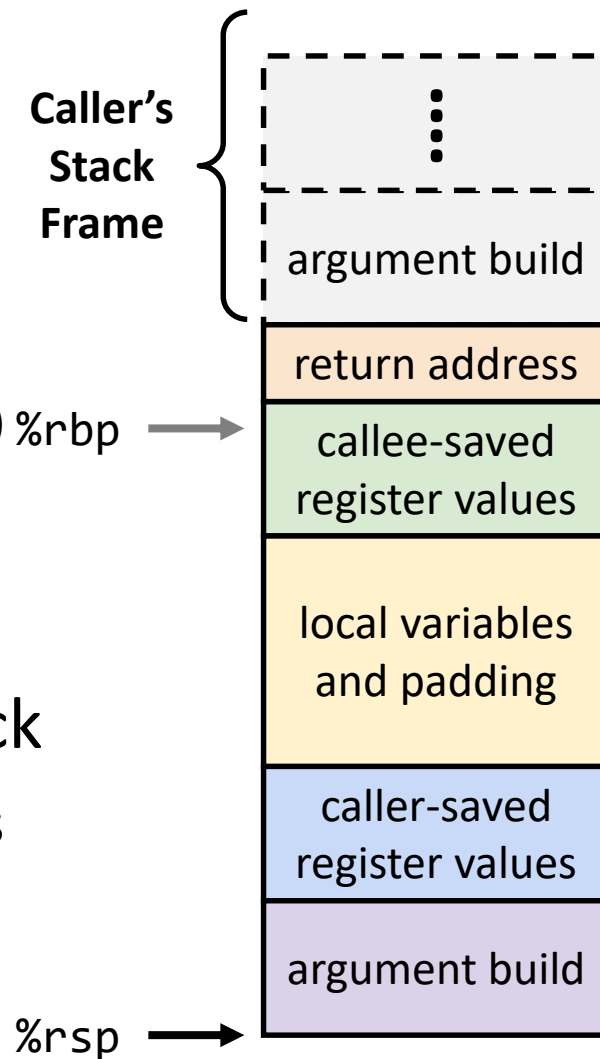
Lesson Summary (1/3)

❖ Important Points

- Procedures are a **combination of *instructions* and *conventions***
 - Conventions prevent functions from disrupting each other
- Stack is the right data structure
 - “Last in, first out” matches lifetime of procedures *(optional) %rbp*
- Recursion handled by normal calling conventions

❖ Generally want to minimize the use of the stack

- Lean heavily on registers, which are faster to access



Lesson Summary (2/3)

%rax	Return value - Caller saved	%r8	Argument #5 - Caller saved
%rbx	Callee saved	%r9	Argument #6 - Caller saved
%rcx	Argument #4 - Caller saved	%r10	Caller saved
%rdx	Argument #3 - Caller saved	%r11	Caller Saved
%rsi	Argument #2 - Caller saved	%r12	Callee saved
%rdi	Argument #1 - Caller saved	%r13	Callee saved
%rsp	Stack pointer	%r14	Callee saved
%rbp	Callee saved	%r15	Callee saved

Lesson Summary (3/3)

- ❖ Terminology:
 - Stack frame: return address, saved registers, local variables, argument build
 - Register saving conventions: callee-saved and caller-saved

- ❖ Learning Objectives:
 - Trace stack frame contents through the execution of x86-64 assembly instructions for both recursive and non-recursive programs.
 - Identify how x86-64 register-saving conventions allow procedures to execute without destroying each other's data.

- ❖ What lingering questions do you have from the lesson?

A detailed, colorful image of a microchip die, showing a complex grid of circuitry and various colored regions (purple, blue, yellow, green, red) representing different functional blocks.

Procedures II – Context

Recursive Example: Popcount

```

/* Recursive popcount */
long pcount_r(unsigned long x) {
  if (x == 0) ← stop once all 1's shifted off
    return 0;
  else
    return (x & 1) + pcount_r(x >> 1);
}

```

logical right shift (pointing to `x >> 1`)

value of LSB (pointing to `x & 1`)

```

pcount_r:
  movl $0, %eax
  testq %rdi, %rdi
  jne .L8
  ret
.L8:
  pushq %rbx
  movq %rdi, %rbx
  shrq %rdi
  call pcount_r
  andl $1, %ebx
  addq %rbx, %rax
  popq %rbx
  ret

```

shift off LSB and recurse (pointing to `shrq %rdi` and `call pcount_r`)

- ❖ Counts the 1's in the binary representation of x
 - <https://godbolt.org/z/P8Mened14>
 - Compiled with `-O1` instead of `-Og` for more natural instruction ordering

- ❖ Register usage:
 - Need x (in `%rdi`) after procedure call
 - Chooses to save `%rdi` by copying into `%rbx`
 - Chooses to save `%rbx` by pushing to stack (only in recursive case)

GDB Demo #2

- ❖ Let's examine the `pcount_r` stack frames on a real machine!
 - Using `pcount.c` from the course website
- ❖ You will need to use GDB to get through the Midterm
 - Useful debugger in this class and beyond!
- ❖ Pay attention to:
 - Checking the current stack frames (`backtrace`)
 - Getting stack frame information (`info frame <#>`)
 - Examining memory (`x`)

A detailed, colorful microchip die image showing intricate circuit patterns in shades of purple, blue, green, and yellow. The text "Procedures II – Practice" is overlaid in white with a drop shadow.

Procedures II – Practice

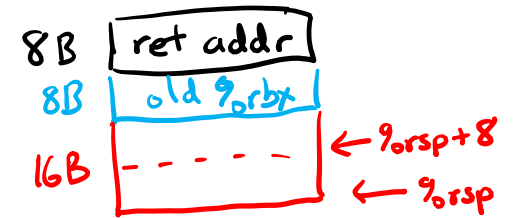
Group Work Time

- ❖ During this time, you are encouraged to work on the following:
 - 1) If desired, continue your discussion
 - 2) Work on the lesson problems (solutions at the end of class)
 - 3) Work on the homework problems

- ❖ Resources:
 - You can revisit the lesson material
 - Work together in groups and help each other out
 - Course staff will circle around to provide support

Practice Questions

- ❖ In the following function, how big is the stack frame?
Which instruction(s) pertain to the local variables and saved registers portions of its stack frame?



```

call_mem_add2:
1  pushq    %rbx           #save a register value
2  subq     $16, %rsp      # allocates space for local variables
3  movq     %rdi, %rbx
4  movq     $351, 8(%rsp)  # initializes local variable value on stack
5  movl     $100, %esi
? 6  leaq    8(%rsp), %rdi  # gets address of local variable (but doesn't actual)
                                use local var
7  call     mem_add
8  addq     %rbx, %rax
9  addq     $16, %rsp      # deallocates space for local variables
10 popq     %rbx          # restore the register value
11  ret

```