Machine-Level Programming III: Procedure

Mechanisms in Procedures

Passing control

- to beginning of procedure code
- back to return point

Passing data

- procedure arguments
- return value

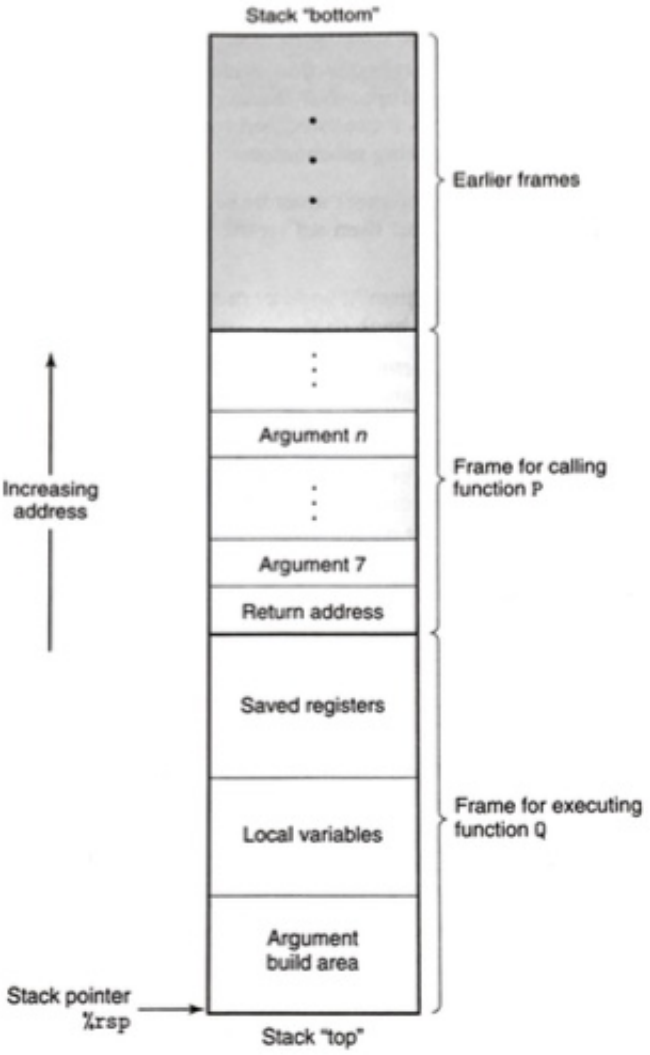
Memory management

- allocate during procedure execution
- deallocate upon return

- mechanisms all implemented with machine instructions
- x86-64 implementation of a procedure uses only those mechanisms required

x86-64 Stack

- region of memory managed with stack discipline
- grows toward lower address
- register `%rsp` contains lowest stack address
    - address of “top” element



Push

`pushq SRC`

- decrement `%rsp` by 8
- write `SRC` at address given by %rsp
- equivalent to
    - `sub $0x8, %rsp`
    - `mov SRC, (%rsp)`

Pop

`popq DEST`

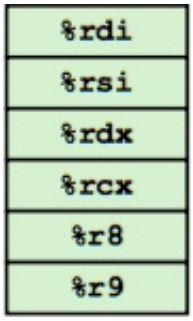
- store value at address given by `%rsp` into `DEST`
- increment `%rsp` by 8
- equivalent to
    - `mov (%rsp), DEST`
    - `add $0x8, %rsp`

Procedure Control Flow

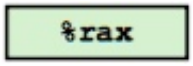
- use stack to support procedure call and return
- **procedure call**: `call label`
    - push return address on stack
    - jump to `label`
- **return address**
    - address of the next instruction right after call
- **procedure return**: `ret`
    - pop address from stack
    - jump to address

Procedure Data Flow

Registers for first 6 arguments



Register for return value



Stack

- only allocate stack space when needed

Stacks and Recursion

- when performing recursion, need some place to store state of each procedure instantiation
    - arguments
    - local variables
    - return pointer
- stack discipline
    - state for given procedure needed for limited time
        - from when called to when return
    - callee returns before caller does
- stack allocated into **frames**
    - state for single procedure instantiation

Stack Frames

- **contents**
    - return information
    - local storage (if needed)
    - temporary space (if needed)
- management
    - space allocated when entering procedure
        - “set-up” code
        - includes push by `call` instruction
    - deallocated when returns
        - “finish” code
        - includes pop by `ret` instruction