


CL 12 

# Procedures

- reusable code fragment that returns a specific task
  - ↳ single named entry pt.
  - ↳ zero or more formal parameters
  - ↳ local storage
  - ↳ return control to the caller when finished

## Implementing Procedures

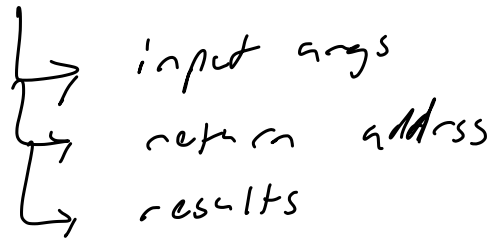
- Option 1: Inlining
  - ↳ compiler substitutes procedure call with body
- Option 2: Linking
  - ↳ produce separate code for each procedure
  - ↳ caller evaluates input arguments, stores them and transfers control to the callee's entry pt.
  - ↳ callee runs, stores results, transfers control to caller

## Calling Convention

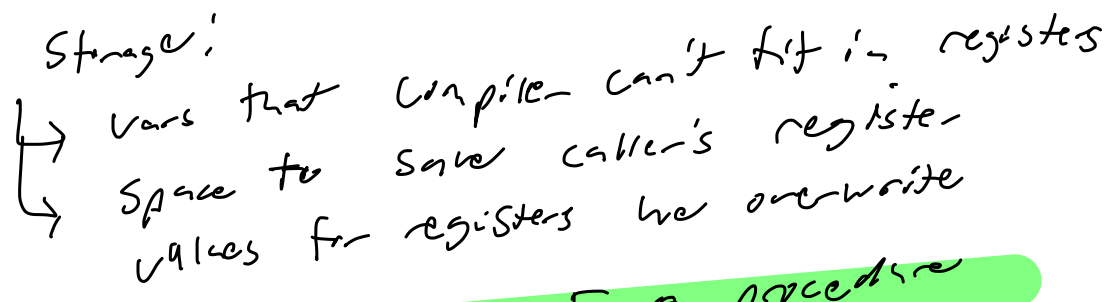
- uniform way to transfer data and control b/w procedures

# Procedure Storage Needs

— basic regs:



— local storage:



\* Specific to the activation of a procedure

# Procedure Linkage: the contract

## Caller will:

- push args onto stack, in reverse order
- branch to callee, putting return address into LP
- remove args from stack on return

## Callee will:

- perform promised computation, leaving result in RD
- branch to return address
- leave stacked data intact, including stacked args
- leave regs (except RD) unchanged

# Summary

- Each procedure invocation has an activation record
  - Created during procedure call/entry sequence
  - Discarded when procedure returns
  - Holds:
    - Argument values (in reverse order)
    - Saved LP, BP from caller (callee reuses those regs)
    - Storage for local variables (if any)
    - Other saved regs from caller (callee needs regs to use)
  - BP points to activation record of active call
    - Access arguments at offsets of -12, -16, -20, ..
    - Access local variables at offsets of 0, 4, 8, ...
- “Callee saves” convention: all reg values preserved
- Except for R0, which holds return value