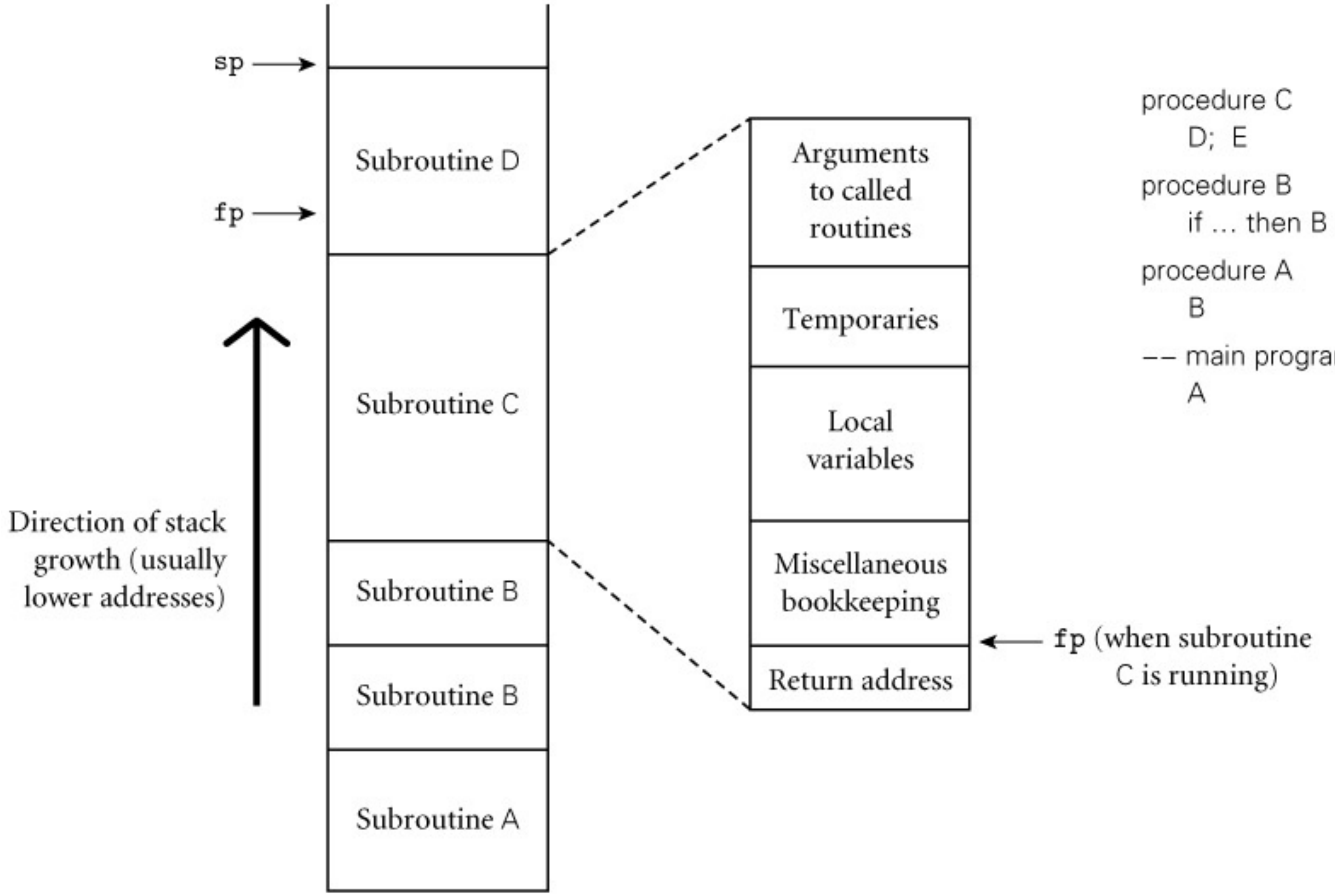
Procedures and Functions

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

Stack-Based Allocation

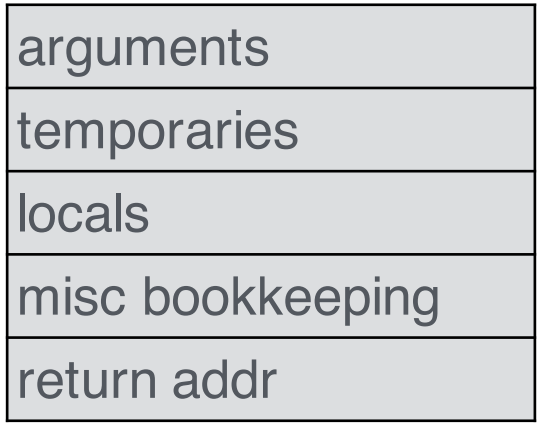

sp
fp
Subroutine D
Subroutine C
Subroutine B
Subroutine B
Subroutine A
Direction of stack growth (usually lower addresses)
Arguments to called routines
Temporaries
Local variables
Miscellaneous bookkeeping
Return address
fp (when subroutine C is running)

```
procedure C
    D;  E
procedure B
    if ... then B else C
procedure A
    B
-- main program
    A
```

Stack Frame (Activation Record)

A stack frame contains:

- Local variables
- Temporaries
- Arguments, return values
- Bookkeeping info

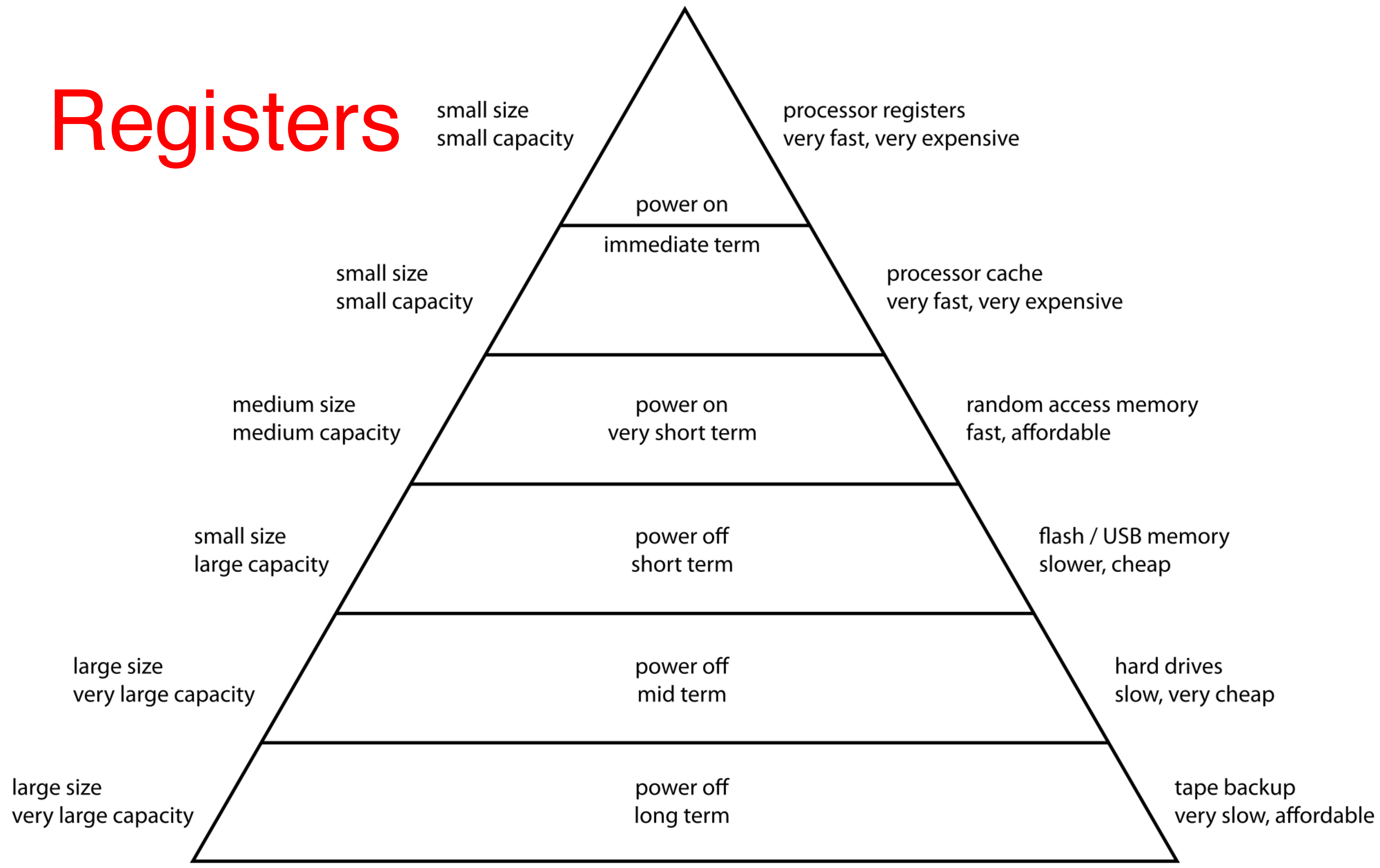


When is it created? By whom?

When is it destroyed? By whom?

Computer Memory Hierarchy

Registers



X86 Registers

General registers
eax, ebx, ecx, edx

Indexes and pointers
sp, pc, fp, …

And many more …

Registers are shared among function calls

Registers

How can registers be used?

When should they be used?

Who saves registers?

What needs to be done before P calls Q?

- Saving dynamic link (current fp)
- Saving other registers
- Changing stack and frame pointer

What needs to be done before Q returns to P?

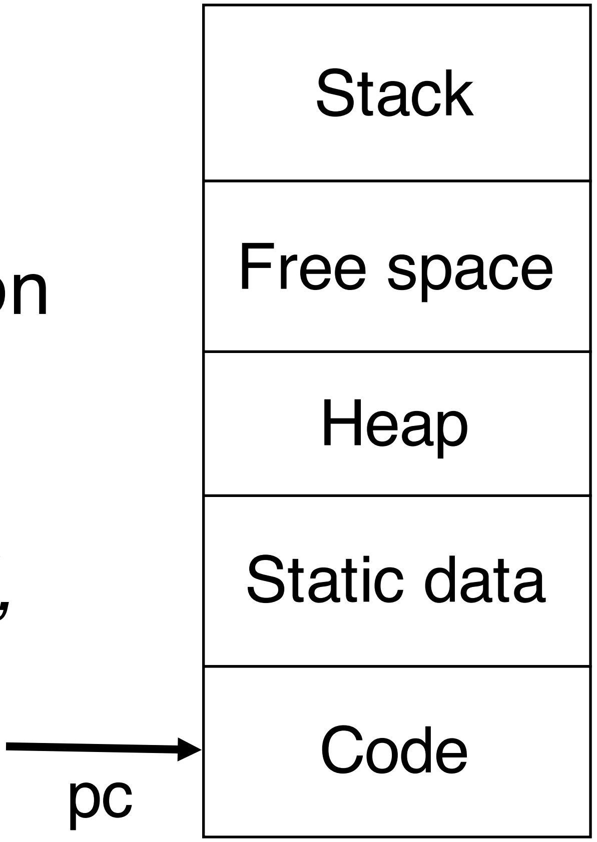
- Restoring saved registers
- Deallocating stack space of Q

Control Flows in HW

Program Counter (PC): a register that always contains the memory address of an instruction

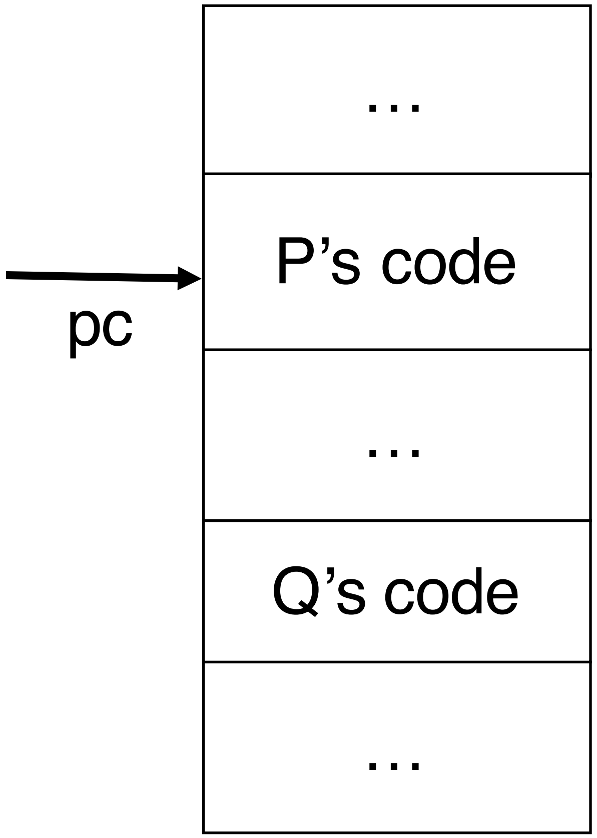
PC always points to the instruction following the one being executed

PC can be modified by *jump, call, return* instructions



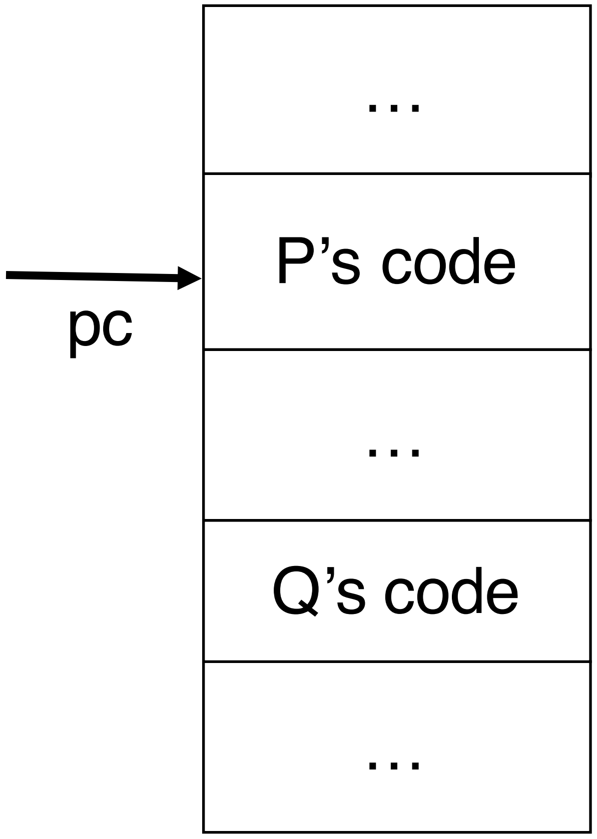
What needs to be done before P calls Q?

- Saving dynamic link (current fp)
- Saving other registers
- Changing stack and frame pointer
- **Saving return address**
- **Changing program counter**



What needs to be done before Q returns to P?

- Restoring saved registers
- Deallocating stack space of Q
- **Restoring program counter**



Parameter Passing

Caller passes parameters to callee

Callee passes values to caller

Formal vs. actual parameters

Parameter passing modes

Value or reference?

Input or output?

Read or write?

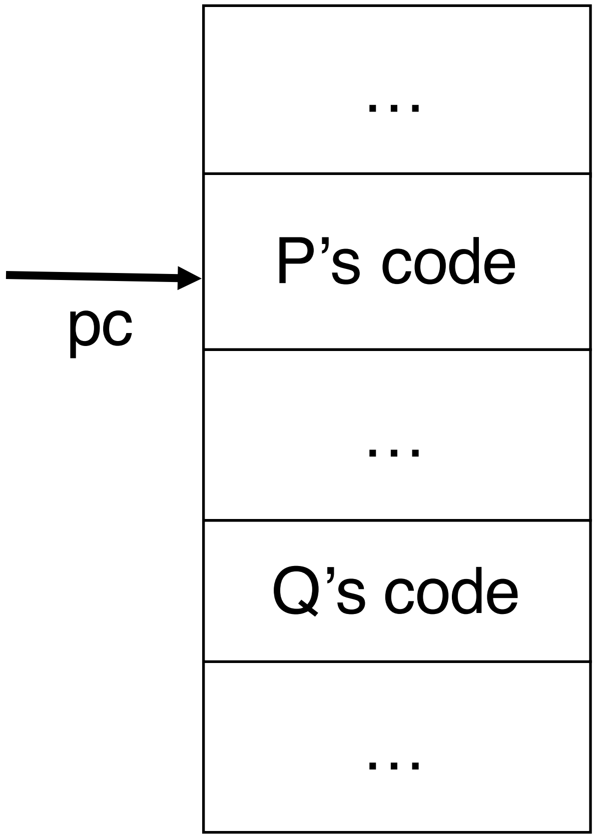
How is parameter passing implemented?

(covered in the next lecture)

What needs to be done before P calls Q?

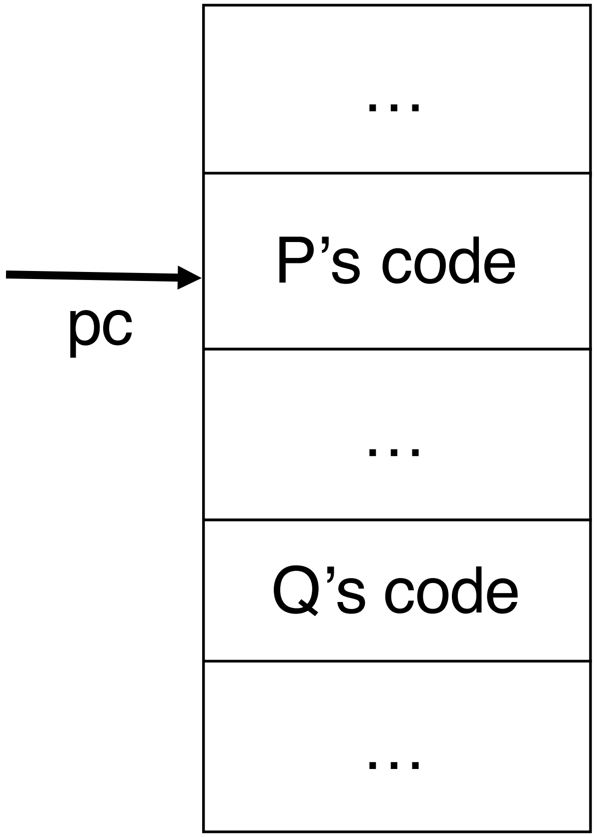
- Saving dynamic link (current fp)
- Saving other registers
- Changing stack and frame pointer
- Saving return address
- Changing program counter
- **Passing parameters**

Who is responsible?



What needs to be done before Q returns to P?

- Restoring saved registers
- Deallocating stack space of Q
- Restoring program counter
- **Passing return values**



Who is responsible?

One Example (implementation dependent)

- Caller processes actual parameters (evaluation, address calculation) and stores them
- Caller stores some control information (e.g., return address) and registers
- Control transferred from Caller to Callee
- Callee allocates storage for locals
- Callee executes
- Callee deallocates storage for locals
- Callee stores return value (if function)
- Control transferred back to Caller
- Caller deallocates storage used for control information and actual parameters
- Caller restores registers

A Typical Calling Sequence

Caller

saves registers
computes values of parameters, move them to stack or registers
switch control to callee (call)

Callee Prologue

allocates frame (change sp)
saves previous frame pointer (dynamic link) and sets new one
saves registers might be overwritten in the callee

A Typical Calling Sequence

Callee Epilogue

moves return value into stack or register
restore callee-saves registers
restore fp and sp
transfer control to caller

Caller

moves return value where needed
restores caller-saves registers when needed

Function call is space and time consuming

Space: each active function requires one frame (activation record) on stack

Time: instructions are added by the compiler to set up and clean up function calls (calling sequence)