Chapter 04: Instruction Sets and the Processor organizations

Lesson 14

Use of the Stack Frames

Objective

- To understand call and return, nested calls and use of stack frames

Calling a subroutine

Calls to the subroutines

- Allow commonly used functions to be written once and used whenever they are needed, and provide abstraction, making it easier for multiple programmers to collaborate on a program
- Important part of virtually all computer languages
- Function calls in a C language program
- Also included additional functions from library by including the header files

Difficulties when Calling a subroutine

Difficulties involved in implementing subroutine calls

1. Programs need a way to pass inputs to subroutines that they call and to receive outputs back from them

Difficulties involved in implementing subroutine calls

2. Subroutines need to be able to allocate space in memory for local variables without overwriting any data used by the subroutine-calling program

Difficulties involved in implementing subroutine calls

3. Since subroutines may be called from many different locations within a program and are often compiled separately from the program that calls them, it is generally impossible to determine which registers may be safely used by a subroutine and which contain data that will be needed after subroutine completes

Difficulties involved in implementing subroutine calls

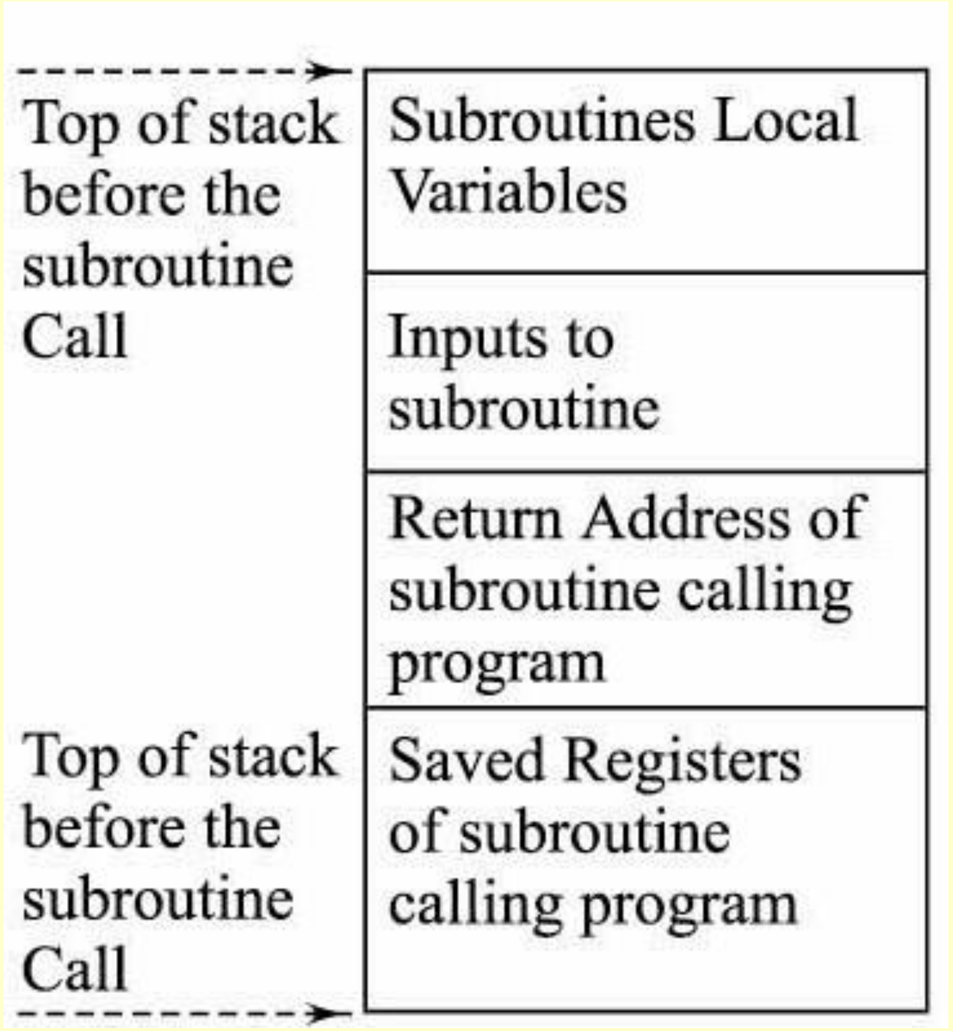
4. Subroutines need a way to figure out where they were called from so execution can return to the calling program when the subroutine completes

A subroutine’s stack frame

1. Space for the contents of the calling program’s register file
2. A pointer to the location that the subroutine should branch to when it completes (calling program return address)
3. The input arguments to the subroutine, and
4. The subroutine’s local variables

Stack frame for a called subroutine

Stack frame


Top of stack before the subroutine Call
Subroutines Local Variables
Inputs to subroutine
Return Address of subroutine calling program
Top of stack before the subroutine Call
Saved Registers of subroutine calling program

Use of the stack frames

- When a subroutine is called, the contents of the calling program's register file are copied into the stack frame, along with its return location and the inputs to subroutine
- The subroutine then uses the rest of the stack frame to hold its local variables

Different subroutines' stack frames of different sizes

- The number of input arguments and local variables varies from subroutine to subroutine
- The arrangement of data within the stack frame also varies from programming system to programming system

Use of stack frame on Return

- When a subroutine finishes, it jumps (returns to the return-address at the stack frame
- Execution of the calling program resumes
- The calling program then reads its saved register file contents out of the stack frame and handles the subroutine's result, which can be passed either in a register or on the stack

Use of stack frame on Return

- Finally, the top-of-stack pointer restored to its position before the subroutine was called, popping the stack frame off of the stack

Nesting of calls using Stack frames

Nesting

- A return from called program is always to its calling program
- The stack structure best suited as a stack functions as LIFO (last in last out) structure

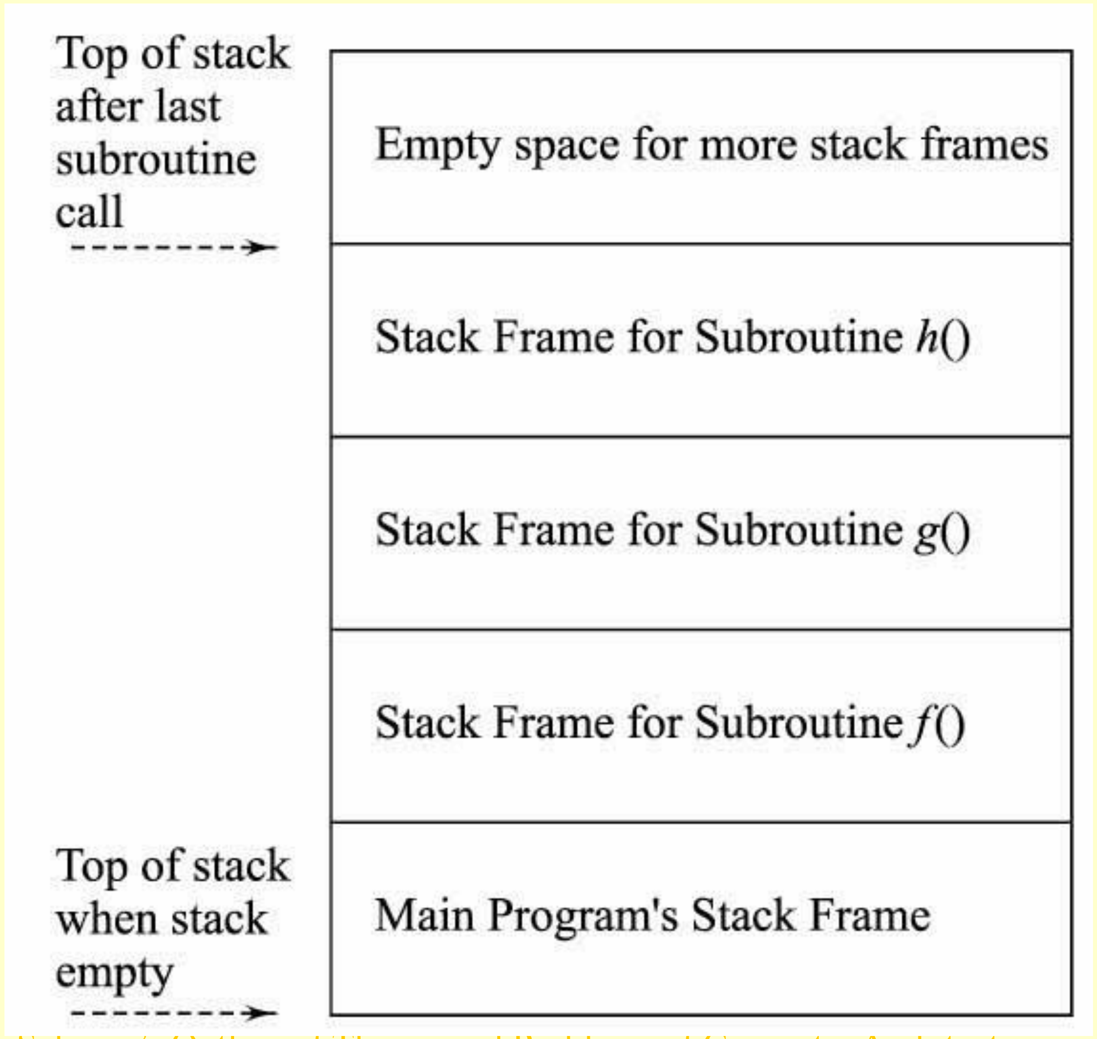
Examples of nesting

- Nesting in main program of f () and g ()
- Note that when g () call f (), f () first returns to g () and when the main program calls f () then f () returns to the main

Program making nested subroutine calls (subroutines that call other subroutines)

- Each nested subroutine allocates its stack frame on top of those already on the stack

The contents of the stack during the execution of subroutine h ()


Top of stack after last subroutine call
Empty space for more stack frames
Stack Frame for Subroutine *h*()
Stack Frame for Subroutine *g*()
Stack Frame for Subroutine *f*()
Top of stack when stack empty
Main Program's Stack Frame

The contents of the stack during the execution of subroutine h ()

- Rhe execution of subroutine h () called from within subroutine g ()
- Subroutine g () was called from within f (), which was called by the main program
- As long as the stack does not overflow, subroutine calls can be nested as deeply as necessary and each stack frame will be popped off of the stack when execution returns to its calling program

Stack overflows

Stack overflow

- The top of stack entering into instruction’s locations of the programs in memory and has become so deep from S0 (empty stack top) that no space is available for further stacking

Calling convention in a programming system

Calling convention

- Defines the requirements that a programming system places on how a subroutine is called and how data is passed between a calling-program and its subroutines
- Used their calling convention to reduce the amount of data that needs to be copied to and from the stack during a subroutine call

Example of a calling convention

- Might specify a set of registers that are used to pass inputs and outputs between the calling program and subroutine

Calling convention

- Different programming systems may arrange the data in a subroutine's stack frames differently
- May require that the steps involved in calling a subroutine be performed in different orders

Summary

We learnt

- Stack frame enables many parameters to be made available to the called routines
- Stack frame of different subroutines can be stacked at the stack
- When a subroutine finishes, it jumps (returns to the return-address at the stack frame and Execution of the calling program resumes

We learnt

- The calling program then reads its saved register file contents out of the stack frame and handles the subroutine's result, which can be passed either in a register or on the stack
- A calling convention used to define the requirements that a programming system places on how a subroutine is called and how data is passed between a calling-program and its subroutines

End of Lesson 14 on

Use of the Stack Frames