Computer Systems Organisation

Tutorial 4

Prepared by: Mehul, Anish

Stack

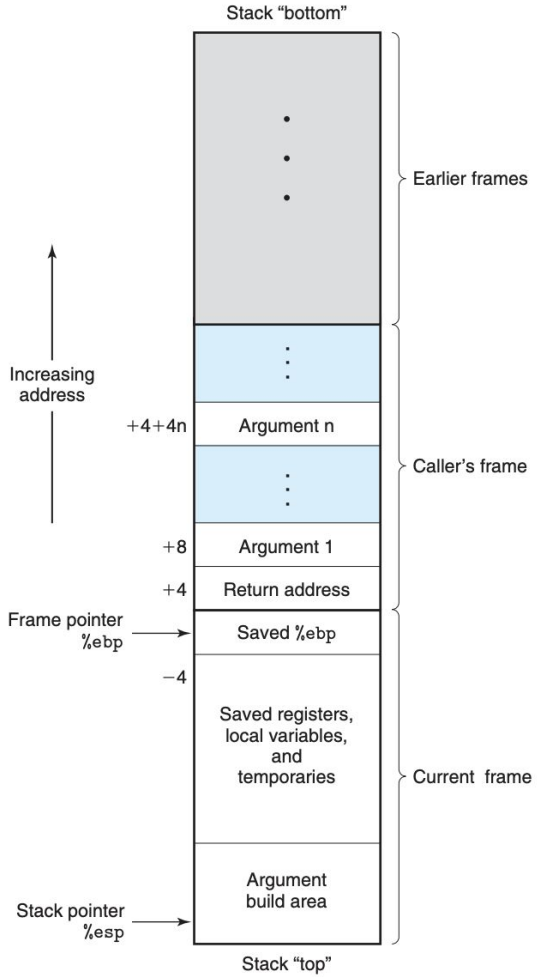
A **stack** is a special area of computer's memory which stores temporary variables created by a procedure. In **stack**, variables are declared, **stored** and initialized during runtime. It is a temporary storage memory.

Space for data with no specified initial value can be allocated on the stack by simply decrementing the stack pointer by an appropriate amount. Similarly, space can be deallocated by incrementing the stack pointer.

Stack instruction:-

- push Reg - Push value in Reg into the stack.
- pop Reg - Pop the top value from the stack in Reg

Figure 3.21
Stack frame structure. The stack is used for passing arguments, for storing return information, for saving registers, and for local storage.



Simple code example

Procedure

- A procedure call involves passing both data (in the form of procedure parameters and return values) and control from one part of a program to another.
- It must also allocate space for the local variables of the procedure on entry and deallocate them on exit.
- Passing of data and the allocation and deallocation of local variables is handled by manipulating the program stack.

Values stored on program stack

- Arguments for callee.
- *Return address* of caller where the program should resume execution when it returns from callee is pushed onto the stack, forming the end of caller's stack frame.
- The callee also uses the stack for its own local variables. Stack also used when the callee acts as caller for any other procedure.

- Register %rbp serve as the frame pointer, and register %rsp serve as the stack pointer.
- The stack grows toward lower addresses and the stack pointer %rsp points to the top element of the stack.

Transferring Control

- call label - The call instruction pushes a return address on the stack and jump to the start of the called procedure. The return address is the address of the instruction immediately following the call in the program.
- ret - The ret instruction pops an address off the stack and jumps to this location.
- leave - The leave instruction can be used to prepare the stack for returning. It is equivalent to the following code sequence:
  - movl %rbp, %rsp - Set stack pointer to beginning of frame
  - popl %rbp - Restore saved %rbp and set stack ptr to end of caller's frame

Register Usage Conventions

- The set of program registers acts as a single resource shared by all of the procedures. Although only one procedure can be active at a given time, we must make sure that when one procedure (the caller) calls another (the callee), the callee does not overwrite some register value that the caller planned to use later.
- Registers %rax, %rdx, and %rcx are classified as caller-save registers. The callee can overwrite these registers without destroying any data required by caller. These can also be used to return values to the caller.
- Registers %rbx, %rsi, and %rdi are classified as callee-save registers. The callee must save the values of any of these registers on the stack before overwriting them, and restore them before returning.

Procedure Example

Recursive Procedure

The stack and linkage conventions allow procedures to call themselves recursively. Since each call has its own private space on the stack, the local variables of the multiple outstanding calls do not interfere with one another. Furthermore, the stack discipline naturally provides the proper policy for allocating local storage when the procedure is called and deallocating it when it returns.

For detailed explanation refer to section 3.7 of the book Computer Systems: A Programmer's Perspective.

Class Exercise

Write a program to recursively calculate n factorial.