15-213 Recitation 4 - 2/12/01

Outline

- Stacks & Procedures
- Jump Tables
- Homogenous Data
- Using gdb

Shaheen Gandhi

e-mail:

sgandhi@andrew.cmu.edu

Office Hours:

Wednesday 1:30 – 2:30

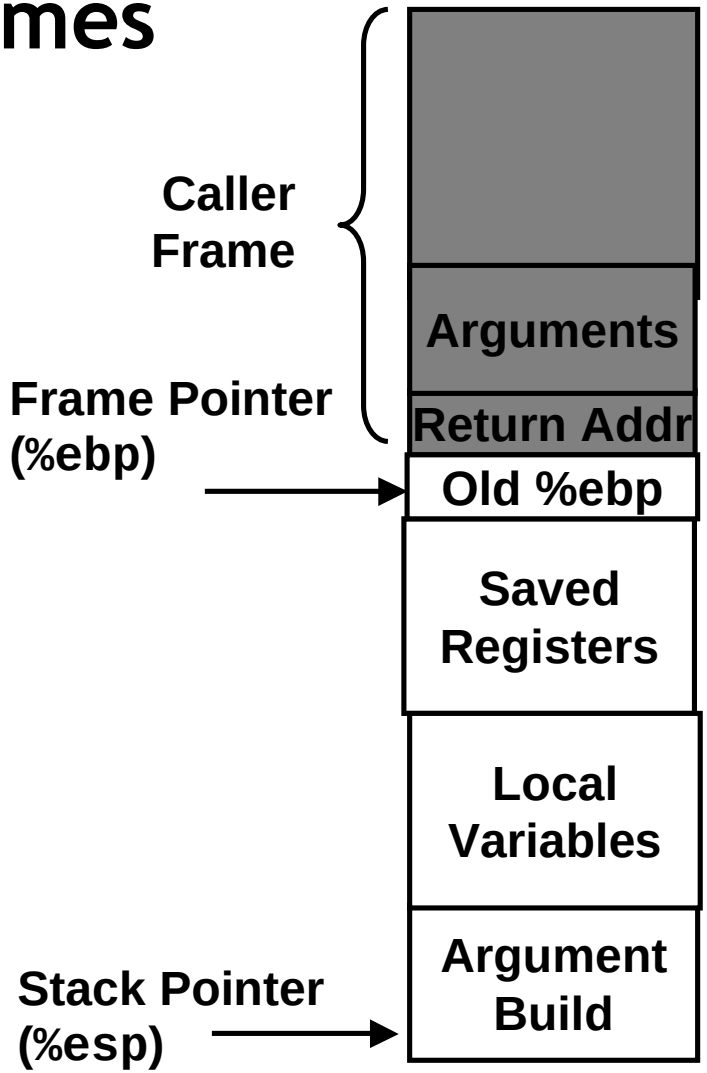
Wean 3108

Stacks

- Grows down
- Stores local variables that can't fit in registers
- Stores arguments and return addresses
- `%esp` Stack Pointer
  - Points to the top value on the stack
- `%ebp` Base Pointer
  - Points to a function's stack frame
- `pushl`
  - Decrements, then places value
- `popl`
  - 'Returns' value, then increments

Stack Frames

- Abstract partitioning of the stack
- Each Frame contains the state for a single function instant



Procedures

`call`: Caller Responsibilities

- Arguments (`pushl`)
  - In what order?
- Return Address (done by `call`)

`ret`: Callee Responsibilities

- Save Registers (especially `%ebp`)
- Set up Stack Frame
- Return value in `%eax`

Jump Tables

- A table of memory addresses
  - Pointers to code blocks that do specific tasks
- Used when many conditionals become inefficient
- Using a Jump Table:
  - `cmpl` one value against another (usually a constant)
  - Use the result to index the jump table (like an array)
  - Retrieve the entry, `jmp` to it

Homogenous Data: Arrays

- Allocated as contiguous blocks of memory

Address Computation Examples

- `int cmu[5] = {…}`
- `cmu` begins at memory address 0x40

```
cmu[0]          40 + 4*0 = 40
cmu[3]          40 + 4*3 = 52
cmu[-1]         40 + 4*-1 = 36
cmu[15]         40 + 4*15 = 100
```