Procedures

Stack frame - The portion of the stack allocated for a single procedure call.

Frame pointer

Stack pointer

Stack "bottom"
1st arg is positioned at offset 8 relative to %ebp.

Arg i is at offset 4 + 4i relative to %ebp.

i = 1, 2, ...

For args ≥ 4 bytes, larger regions on stack is used.
Transferring Control

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Desc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>call Label</td>
<td>Procedure Call</td>
</tr>
<tr>
<td>call xOperand</td>
<td>Prepare stack for return.</td>
</tr>
<tr>
<td>leave</td>
<td>Return from call.</td>
</tr>
<tr>
<td>ret</td>
<td></td>
</tr>
</tbody>
</table>

**call**
1. push a return address on the stack.
2. jump to the start of the called procedure.

**return address** - address of the instruction immediately following the call.

**ret**
1. pops an address off the stack.
2. jumps to that address.
leave - prepare the stack for returning.

movl %ebp, %esp

popl %ebp

set stack pointer to beginning of frame.

Restore saved %ebp and set stack ptr to end of caller's frame.

Register Usage Conventions

Caller - save registers - %eax, %edx, %ecx.

main( ) (caller) must save these registers.

main( ) - caller

↓

sum( ) - callee.

Callee - save registers - %ebx, %esi, %edi

sum( ) (callee) must save these registers.

eg. int p(int x)

1. int y = x + x;
2. int z = Q(y);
3. return y + z;

1. y can be saved in p's stack frame.
2. y can be saved in a callee-save register, in which case Q must make sure that it should save & restore the value of y.