int multiply (inta, intb) 
{
    return a*b;
}

int caller ()
{
    int i = 12;
    int j = 15;
    multiply();
    return values;
    save/restore registers used by caller
    stack in case we run out of registers
}

STACK
Every function or procedure has a stack frame

Top of the stack hold the most recent value

$\Rightarrow$ confusing!

actually lowest address

Stack is delimited by 2 points

$\%EBP \Rightarrow$ base of current frame

$\%ESP \Rightarrow$ top of current frame
new instructions

know push and pop

Subl $4, %esp
Movl %esp, (%esp)
amd %esp

movl (%esp), %d

Call label

2 things 1) Pushes return address onto the stack
2) jumps to label

0x1000 -->
0x1002 - call func
0x1006 movl --

ret => returns
opposite of call
pops the top item on stack
jumps to that address
leave called before return

-> fixes up the stack back to original value

movl %ebp, %esp
popl %ebp

---

Return value always in %eax

---

What to do with reused registers?

movl $12, %edx
movl $20, %esi
call cool_func

movl $5000, %edx
movl $0, %esi

2 options:

**Caller saves registers**

**callee saves**
Some saved by caller \rightarrow \textbf{Volatile}
\%eax, \%edx, \%ecx

Some are callee saved \rightarrow \textbf{Non-Volatile}
\%ebx, \%esi, \%edi