

Stack Frame

CSE 361S

Pass Parameters

- Caller pushes args on stack, last to first

```
foo (int a, int b) {
    ... a+b ...
}

main: ...
    push b          /* push args in reverse order */
    push a
    call foo
    addl $8, %esp  /* one inst. to deallocate args */
```

Reference Parameters

- Parameters referenced using frame pointer:

```
foo: ...
    movl 8(%ebp), %edx /* %edx ← a */
    addl 12(%ebp), %edx /* %edx ← a + b */
```

Frame Pointer (%ebp)

- Maintaining the frame pointer is the callee's job

```
proc: pushl %ebp          /* adjust frame */
      movl    %esp, %ebp
      ...
      movl    %ebp, %esp /* restore frame */
      popl    %ebp
      ret
```

Stack Figure

Local Variables

- Callee allocates stack space for locals

```
bar (int a) {
    int x, y, ...;
    ...
    x = a + y;
    ...
}
```

```

bar:  pushl %ebp      /* adjust frame */
      movl %esp, %ebp
      subl $32, %esp /* allocate space for 8 ints */
      ...

```

```

      movl -8(%ebp), %eax /* %eax ← y */
      addl 8(%ebp), %eax /* %eax ← y + a */
      movl %eax, -4(%ebp) /* x ← y + a */

```

Summary of Responsibilities

- caller
 - save caller regs
 - push args
 - call routine
 - deallocate args
 - restore regs
- callee
 - adjust stack frame
 - allocate local vars
 - save callee regs
 - ...
 - restore regs
 - deallocate local vars
 - restore frame

Returning Values from Functions

- return value in %eax if it fits, if it is > 4 bytes, ptr to value

```

int func (int i) {
    return(i);
}

func:  pushl %ebp
      movl %esp, %ebp
      movl 8(%ebp), %eax
      movl %ebp, %esp
      popl %ebp
      ret

```

Complete Example

```

int add (int x, int y) {
    int sum;
    ...
    m = add(a, b);
    ...
    sum = x + y;
    return(sum);
}

```

```

add:  pushl %ebp      /* adjust stack frame */
      movl %esp, %ebp
      subl $4, %esp /* allocate sum */
      pushl %ebx     /* save regs */
      movl 8(%ebp), %ebx /* x */
      addl 12(%ebp), %ebx /* y */
      movl %ebx, -4(%ebp) /* sum = x + y */
      movl %ebx, %eax /* return value */
      popl %ebx     /* restore regs */
      movl %ebp, %esp /* deallocate locals */
      popl %ebp     /* restore frame */
      ret

```

Complete Example

```

int add (int x, int y) {
    int sum;
    ...
    m = add(a, b);
    ...
    sum = x + y;
    return(sum);
}

```

```

pushl %ecx    /* save regs */
pushl %edx
pushl b      /* push args */
pushla
call add     /* call */
addl $8, %esp /* deallocate args */
popl %edx    /* restore regs */
popl %ecx
movl %eax, m /* m ← add(a, b) */

```

```

*p = d;
return x-c;

```

```

movsbl 12(%ebp), %edx /* 8 bits to 32 bits */
movl 16(%ebp), %eax
movl %edx, (%eax)
movswl 8(%ebp), %eax /* 16 bits to 32 bits */
movl 20(%ebp), %edx
subl %eax, %edx
movl %edx, %eax

```

```

int fun(short c, char d, int *p, int x);

```

```

int fun(short c, char d, int *p, int x){
    *p = d;
    return x-c;
}

```

Vulnerable Code Example

```

int getbuf() {
    char buf[16];

    gets(buf); /* input string until '\r' with
               no bounds checks */
    return 1;
}

```