CMSC 313 Lecture 12

- Project 3 Questions
- How C functions pass parameters

Last Time

- **Stack Instructions: PUSH, POP**
  - PUSH adds an item to the top of the stack
  - POP removes an item from the top of the stack
- **Subroutine Instructions: CALL, RET**
  - CALL saves EIP on the stack and jumps to the subroutine
  - RET retrieves the caller's EIP from the stack
- **Subroutine Examples**

Linux/gcc/i386 Function Call Convention

- Parameters pushed right to left on the stack
  - first parameter on top of the stack
- Caller saves EAX, ECX, EDX if needed
  - these registers will probably be used by the callee
- Callee saves EBX, ESI, EDI
  - there is a good chance that the callee does not need these
- EBP used as index register for parameters, local variables, and temporary storage
- Callee must restore caller's ESP and EBP
- Return value placed in EAX

A typical stack frame for the function call:

```
int foo (int arg1, int arg2, int arg3) ;
```

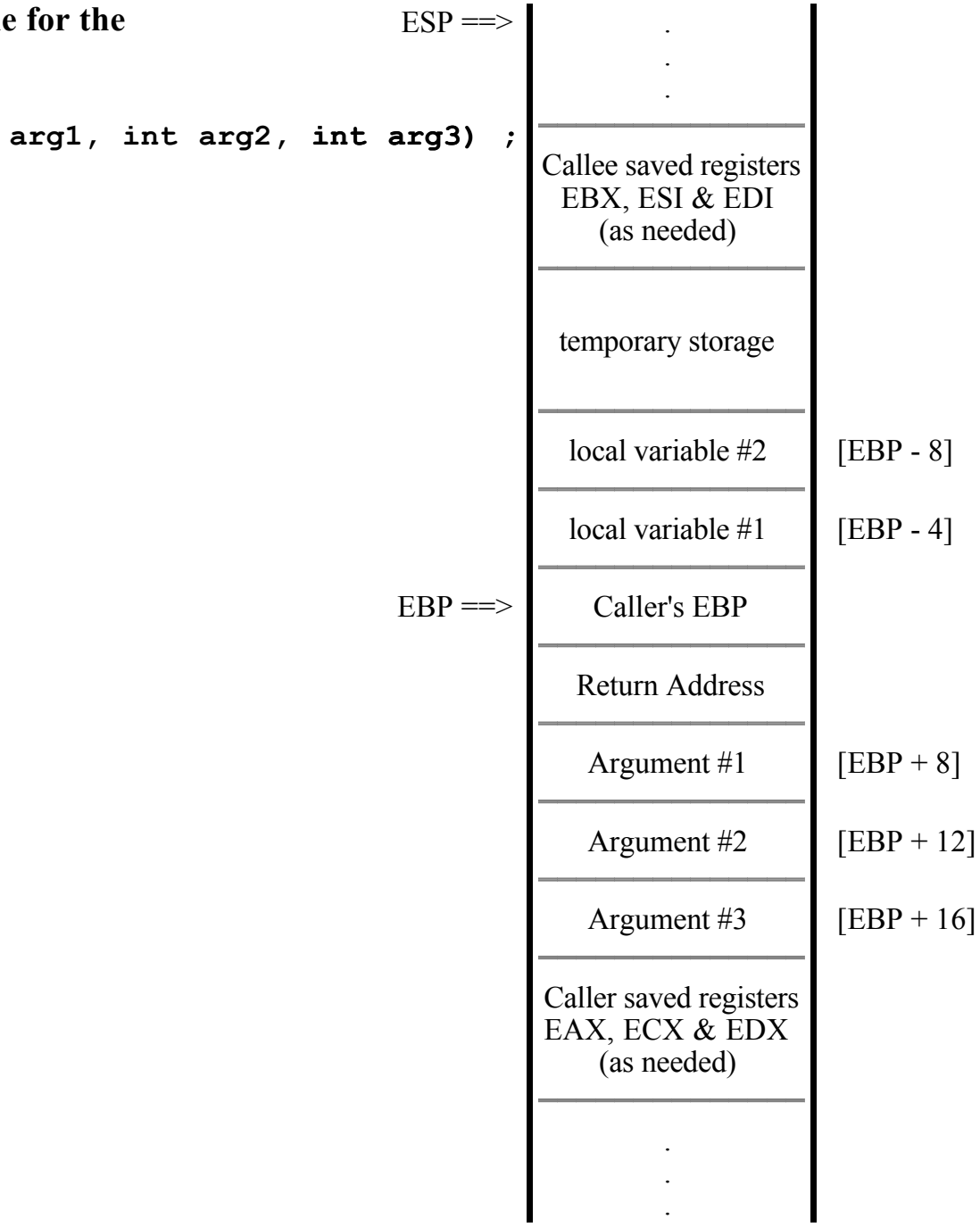

Fig. 1

The caller's actions before the function call

- Save EAX, ECX, EDX registers as needed
- Push arguments, last first
- CALL the function

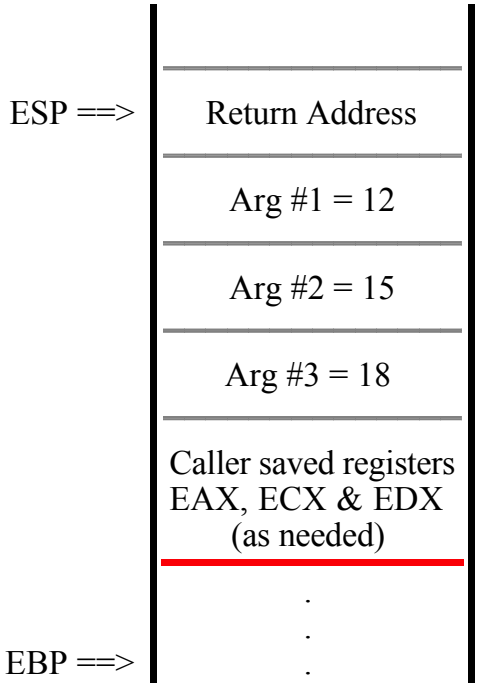

Fig. 2

The callee's actions after function call

- Save main's EBP, set up own stack frame

```
push    ebp
mov     ebp, esp
```

- Allocate space for local variables and temporary storage
- Save EBX, ESI and EDI registers as needed

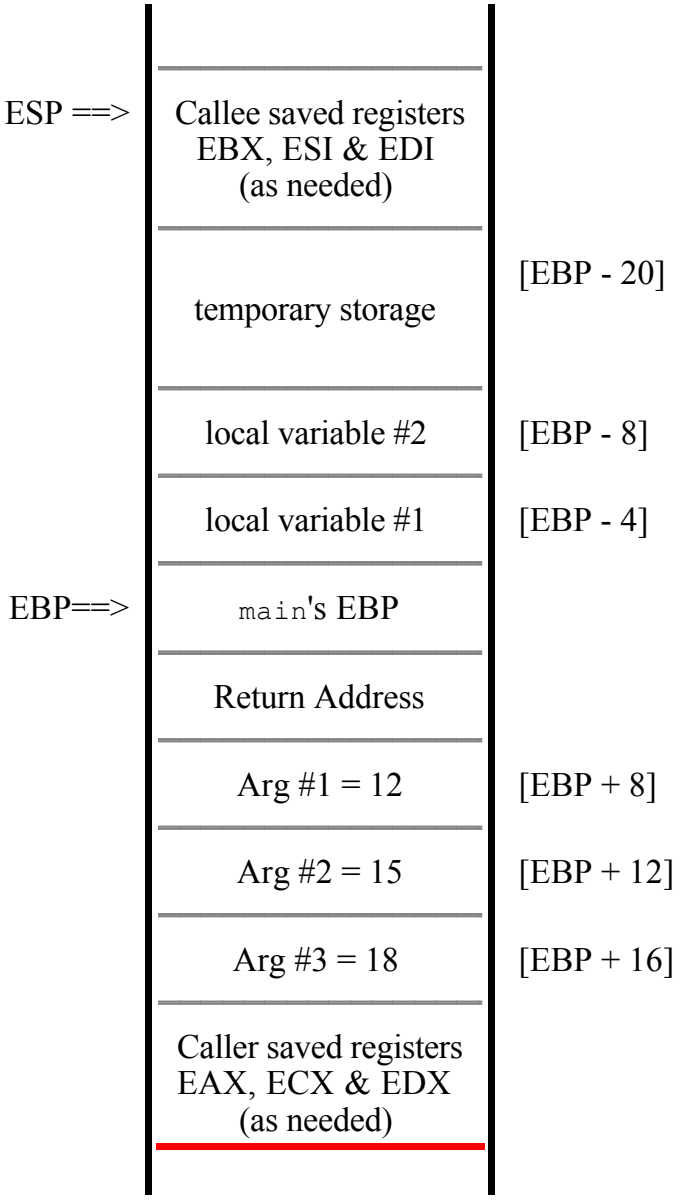

Fig. 4

The callee's actions before returning

- Store return value in EAX
- Restore EBX, ESI and EDI registers as needed
- Restore main's stack frame

```
mov     esp, ebp
pop     ebp
```

- RET to main

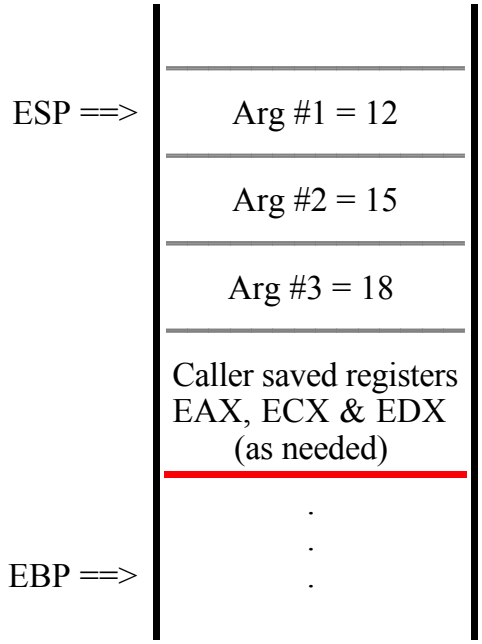

Fig. 5

The caller's actions after returning

- POP arguments off the stack
- Store return value in EAX
- Restore EAX, ECX and EDX registers as needed

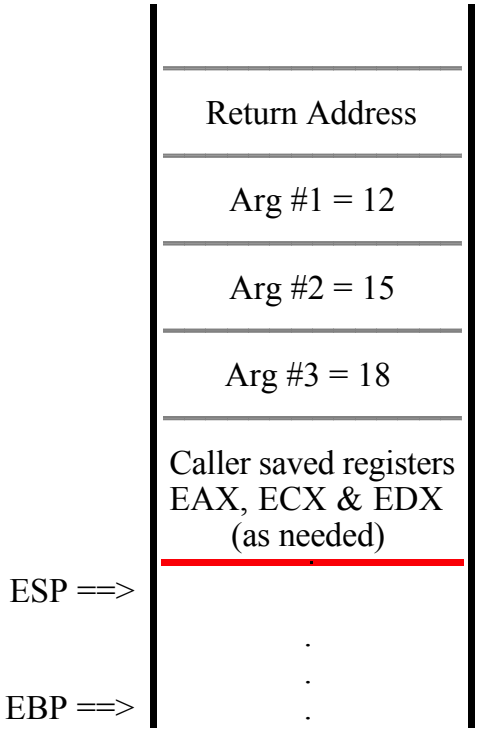

Fig. 6

```
// File: cfunc.c
//
// Example of C function calls disassembled
//

#include <stdio.h>

// a silly function
//
int foo(int x, int y) {

   int z ;

   z = x + y ;
   return z ;
}

int main () {
   int b ;

   b = foo(35, 64) ;
   b = b + b ;
   printf ("b = %d\n", b) ;
}
```

```
linux3% gcc cfunc.c
linux3% a.out
b = 198
linux3%


linux3% gcc -S cfunc.c
linux3% i2g -g cfunc.s >cfunc.asm
linux3%
```

```
        .file   "cfunc.c"
        .version        "01.01"
gcc2_compiled.:
.text
        .align 4
.globl foo
        .type    foo,@function
foo:
        pushl %ebp
        movl %esp,%ebp
        subl $4,%esp
        movl 8(%ebp),%eax
        movl 12(%ebp),%edx
        leal (%edx,%eax),%ecx
        movl %ecx,-4(%ebp)
        movl -4(%ebp),%edx
        movl %edx,%eax
        jmp .L1
        .p2align 4,,7
.L1:
        leave
        ret
```

```
.Lfe1:
        .size    foo,.Lfe1-foo
.section        .rodata
.LC0:
        .string "b = %d\n"
.text
        .align 4
.globl main
        .type    main,@function
main:
        pushl %ebp
        movl %esp,%ebp
        subl $4,%esp
        pushl $64
        pushl $35
        call foo
        addl $8,%esp
        movl %eax,%eax
        movl %eax,-4(%ebp)
        movl -4(%ebp),%eax
        addl %eax,-4(%ebp)
        movl -4(%ebp),%eax
        pushl %eax
        pushl $.LC0
        call printf
        addl $8,%esp
.L2:
        leave
        ret
.Lfe2:
        .size    main,.Lfe2-main
        .ident  "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"
```

```
        ;FILE "cfunc.c"
gcc2_compiled.:
SECTION .text
        ALIGN 4
GLOBAL foo
        GLOBAL foo:function
foo:
        push  ebp
        mov  ebp,esp
        sub  esp,4
        mov  eax, [ebp+8]
        mov  edx, [ebp+12]
        lea  ecx, [edx+eax]
        mov  [ebp-4],ecx
        mov  edx, [ebp-4]
        mov  eax,edx
        jmp L1
        ;ALIGN 1<<4 ; IF < 7
L1:
        leave
        ret
```

```
.Lfe1:
        GLOBAL   foo:function (.Lfe1-foo)
SECTION         .rodata
.LC0:
        db      'b = %d',10,''
SECTION .text
        ALIGN 4
GLOBAL main
        GLOBAL main:function
main:
        push  ebp
        mov  ebp,esp
        sub  esp,4
        push  dword 64
        push  dword 35
        call foo
        add  esp,8
        mov  eax,eax
        mov  [ebp-4],eax
        mov  eax, [ebp-4]
        add  [ebp-4],eax
        mov  eax, [ebp-4]
        push  eax
        push  dword .LC0
        call printf
        add  esp,8
L2:
        leave
        ret
.Lfe2:
        GLOBAL   main:function (.Lfe2-main)
        ;IDENT "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"
```

```
.Lfe1:
        GLOBAL   foo:function (.Lfe1-foo)
SECTION         .rodata
.LC0:
        db      'b = %d',10,''
SECTION .text
        ALIGN 4
GLOBAL main
        GLOBAL main:function
main:
        push  ebp
        mov  ebp,esp
        sub  esp,4
        push  dword 64
        push  dword 35
        call foo
        add  esp,8
        mov  eax,eax
        mov  [ebp-4],eax
        mov  eax, [ebp-4]
        add  [ebp-4],eax
        mov  eax, [ebp-4]
        push  eax
        push  dword .LC0
        call printf
        add  esp,8
L2:
        leave
        ret
.Lfe2:
        GLOBAL   main:function (.Lfe2-main)
        ;IDENT "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"
```

Next Time

- C function call examples
- Virtual Memory