Function Calls and Calling Conventions 2

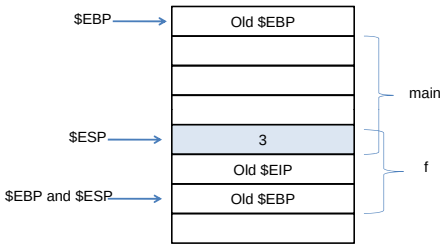
Jonathan Misurda
jmisurda@cs.pitt.edu

Function Call, 1 param

```
#include <stdio.h>

int f(int x)
{
        return x;
}

int main()
{
        int y;

        y = f(3);

        return 0;
}
```

```
f:
        pushl   %ebp
        movl    %esp, %ebp
        movl    8(%ebp), %eax
        leave
        ret

main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        subl    $16, %esp
        movl    $3, (%esp)
        call    f
        movl    %eax, -4(%ebp)
        movl    $0, %eax
        leave
        ret
```

Stack

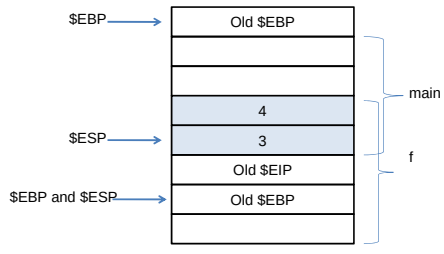


Function Call, 2 params

```
#include <stdio.h>

int f(int x, int y)
{
        return x+y;
}

int main()
{
        int y;

        y = f(3, 4);

        return 0;
}
```

```
f:      pushl   %ebp
        movl    %esp, %ebp
        movl    12(%ebp), %eax
        addl    8(%ebp), %eax
        leave
        ret
main:   pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        subl    $16, %esp
        movl    $4, 4(%esp)
        movl    $3, (%esp)
        call    f
        movl    %eax, 4(%esp)
        movl    $0, %eax
        leave
        ret
```

Stack



Observation

- Parameters are pushed right to left onto the stack
- Why?

printf

```
int printf(const char *format,...);
```

- "…" means variable number of arguments

#include <stdarg.h>

```
int *makearray(int a, ...) {
  va_list ap;
  int *array = (int *)malloc(MAXSIZE * sizeof(int));
  int argno = 0;
  va_start(ap, a);
  while (a > 0 && argno < MAXSIZE) {
      array[argno++] = a;
      a = va_arg(ap, int);
  }
  array[argno] = -1;
  va_end(ap);
  return array;
 }
```

Variable Arguments Usage

```
int main()
{
        int *p;
        int i;
        p = makearray(1,2,3,4,-1);

        for(i=0;i<5;i++)
                printf("%d\n", p[i]);

        return 0;
}
```

Other Notes

- Also called a *Variadic* function
- Old header <varargs.h> no longer favored for use
- C99 Includes support for Variadic Macros
- Java:

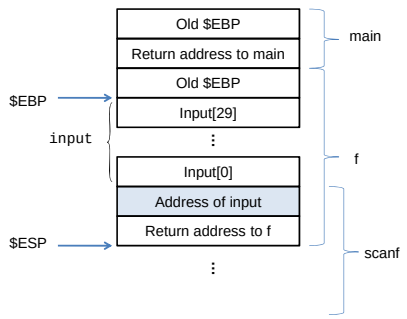
```
public static void printArray(Object... objects) {
  for (Object o : objects)
      System.out.println(o);
}

printArray(3, 4, "abc");
```

Stack Allocated Array

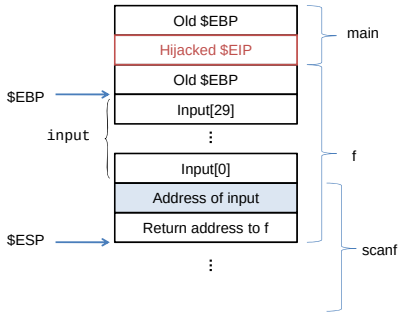
```
void f()
{
  char input[30];
  scanf("%s", input);
}

int main()
{
  f();
  return 0;
}
```

```
f:      pushl   %ebp
        movl    %esp, %ebp
        subl    $56, %esp
        leal    -40(%ebp), %eax
        movl    %eax, 4(%esp)
        movl    $.LC0, (%esp)
        call    scanf
        leave
        ret

main:   pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        subl    $16, %esp
        call    f
        movl    $0, %eax
        leave
        ret
```

Stack



Buffer Overrun



Buffer Overrun Vulnerability