15-213 Recitation 3: 09/23/02

Outline

- Register convention
- Stacks and procedures
- Arrays
- Structs and linked list

Reminder

- L2 is due this Thursday!

Annie Luo

e-mail:

luluo@cs.cmu.edu

Office Hours:

Thursday 6:00 – 7:00

Wean 8402

Registers

- **`%eax:`** Primary accumulator. contains return value from a function call
- **`%ebx:`** Accumulator. Often used to hold the starting address of an array
- **`%ecx:`** Accumulator. Often used as a counter or index register for an array or a loop
- **`%edx:`** General purpose register
- **`%esi:`** General purpose register. Pointer to *source* address when copying a block of data.
- **`%edi:`** General purpose register. Pointer to *destination* address when copying a block of data.

Stack

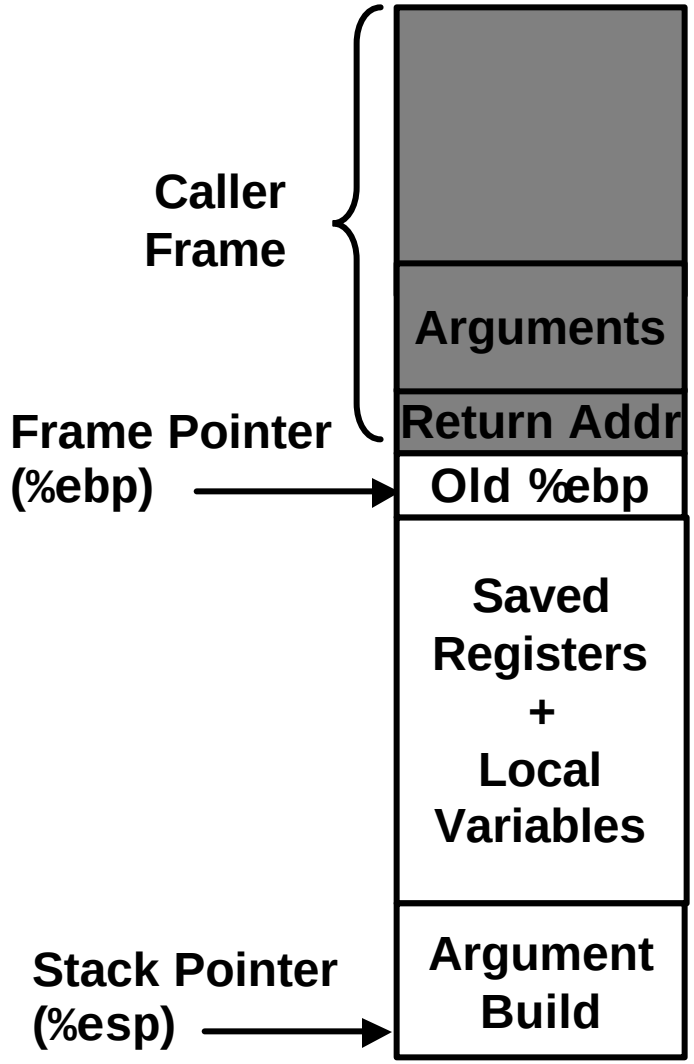
- Makes recursion work
- Grows toward lower addresses

- **%esp** stack “top”, lowest stack address
- **%ebp** stack “bottom”, start of current frame

- Pushing
  - `add $0xfffffffc, %esp  # decrease: %esp-4`
  - `push %eax  # place value`
- Popping
  - `pop %eax  #read value pointed by %esp`
  - `add $0x4, %esp  #increase %esp+4`

Stack Frames

- Temporary space allocated when enter procedure
- Private storage for each *instance* of procedure call



Procedure

`call`: caller responsibilities

- Save *caller save* registers (if necessary)
  - %eax, %ecx, %edx
- Push arguments (in what order?)
- Save return address (done by `call`)

`ret`: callee responsibilities

- Save base pointers

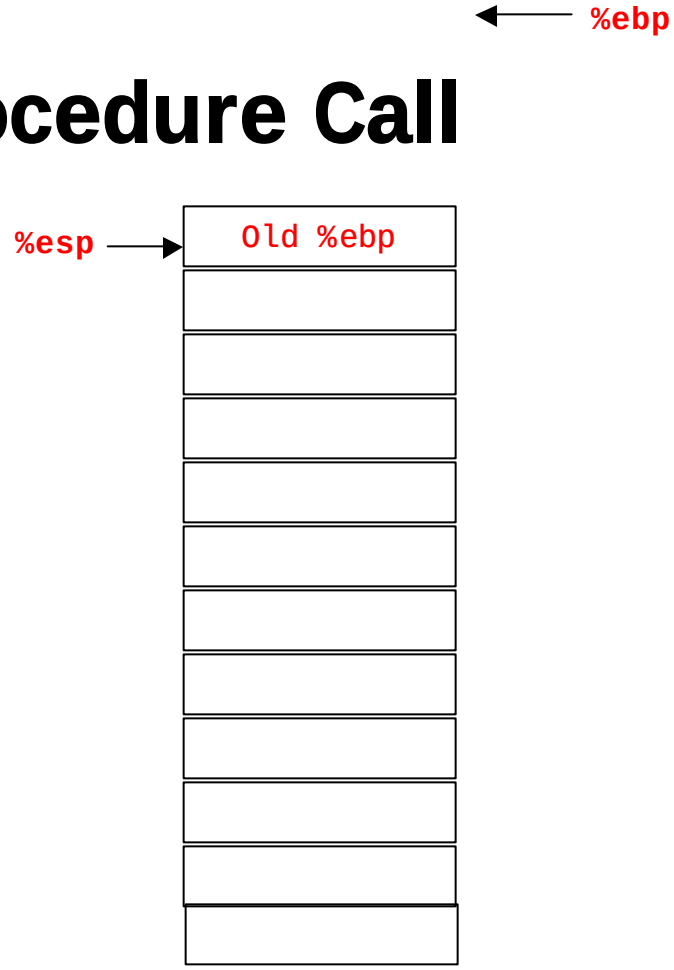
```
push %ebp
mov %esp, ebp
```

- Save callee save registers
  - %ebx, %esi, %edi
- Return value in `%eax`
- Restore registers

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

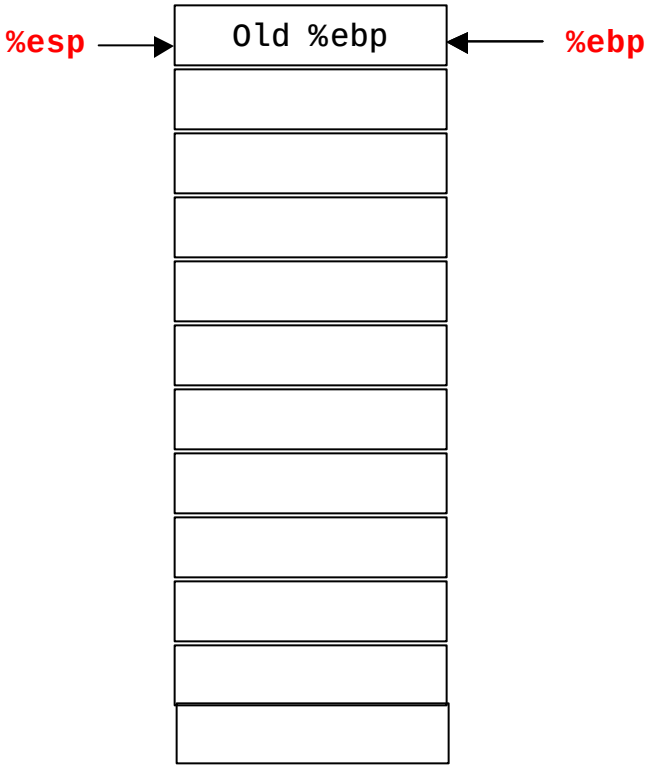
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



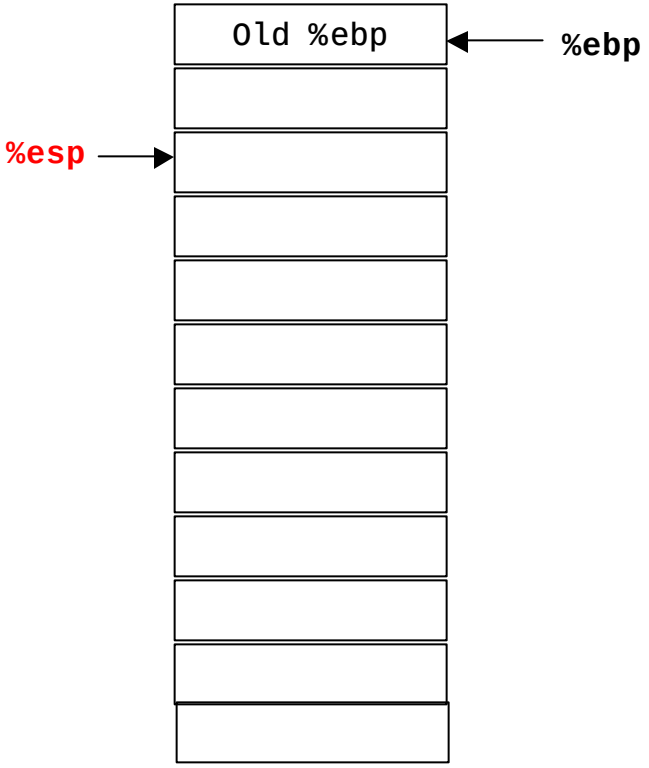
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



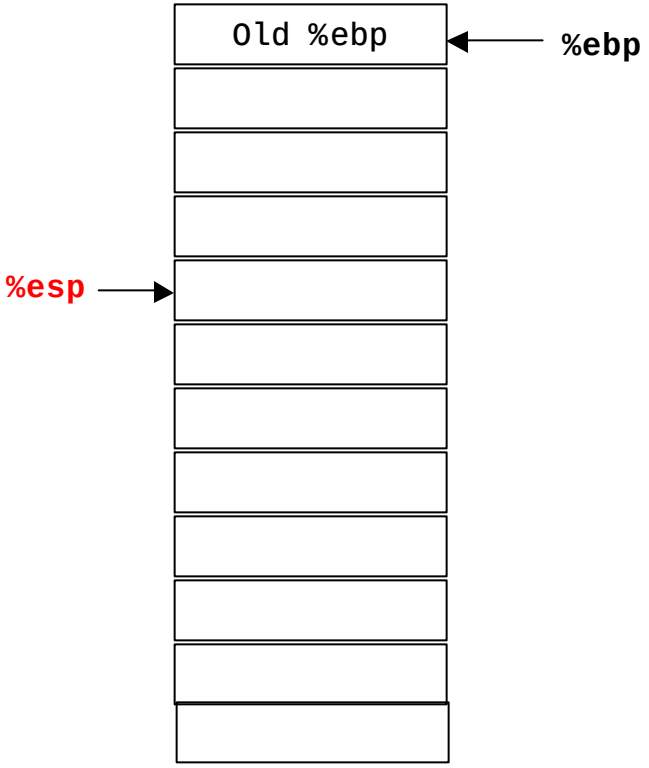
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



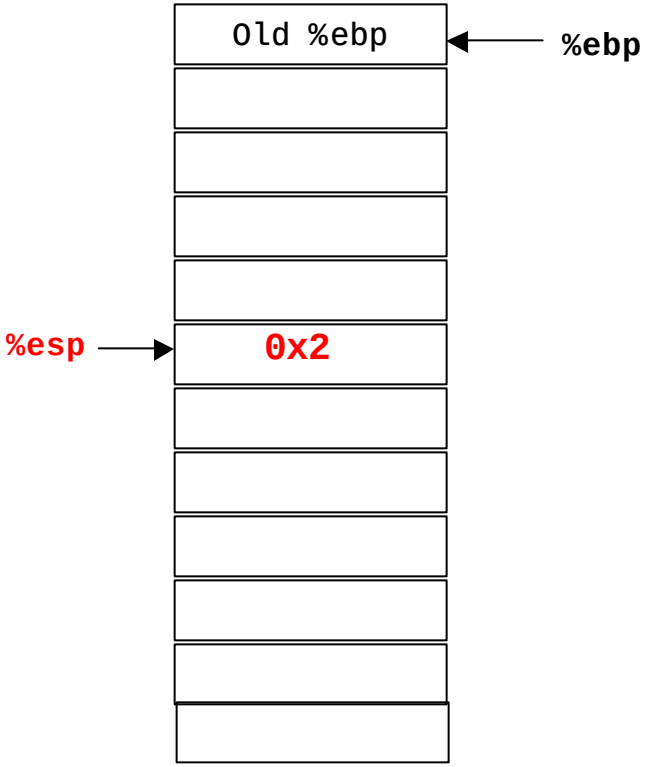
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



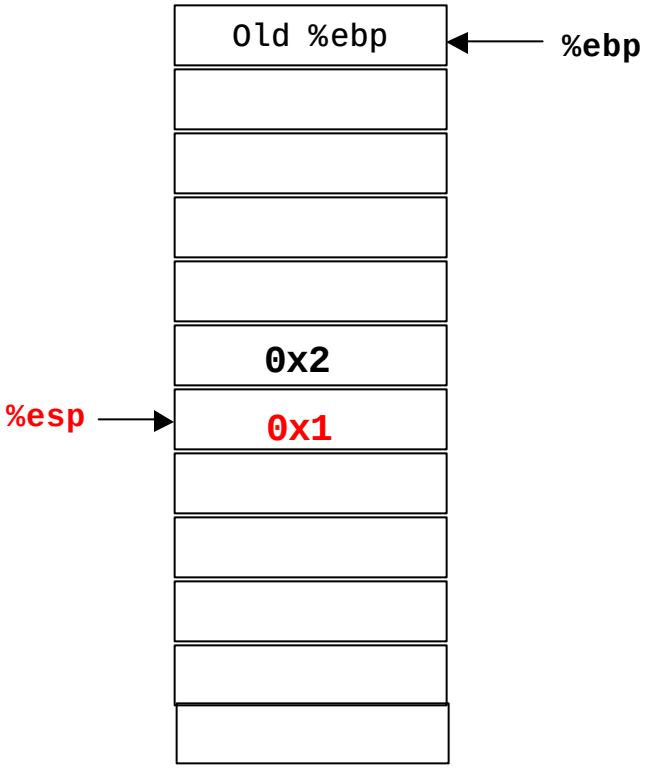
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

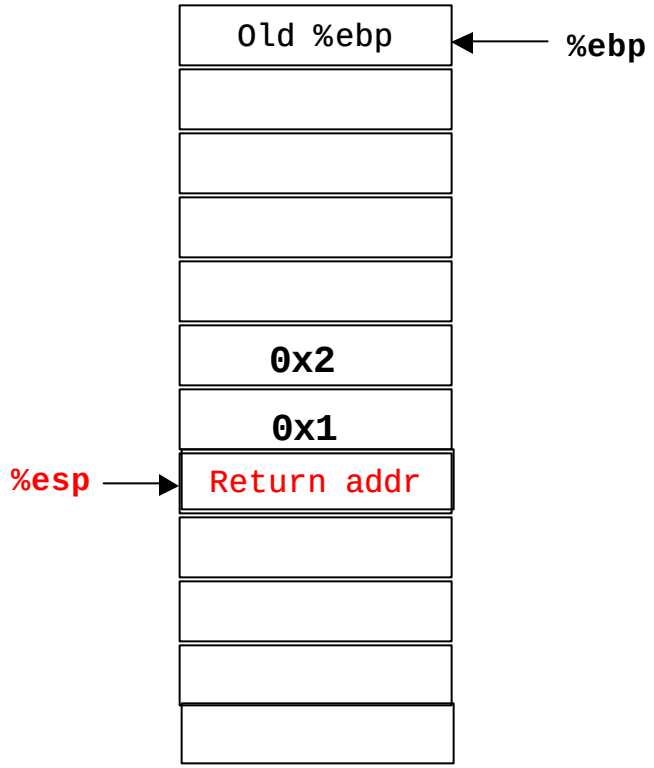
```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

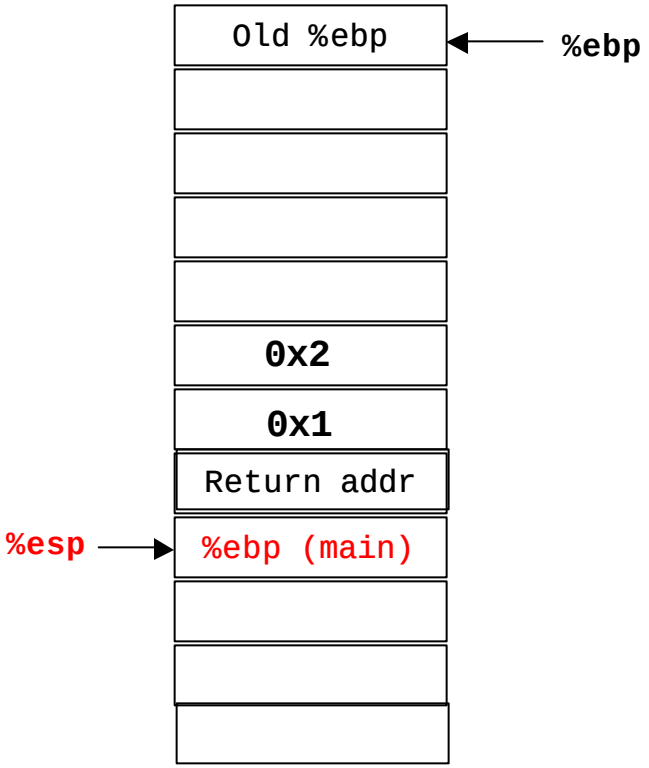
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

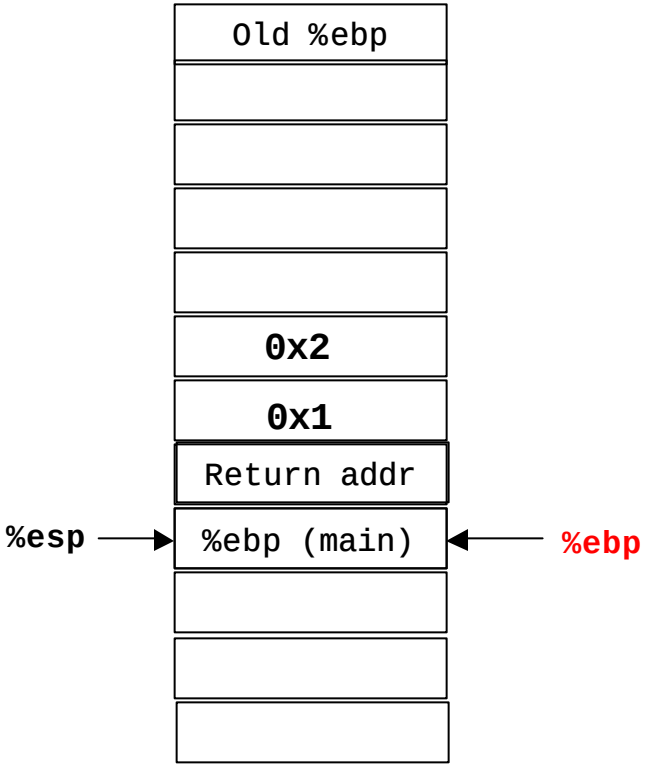
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

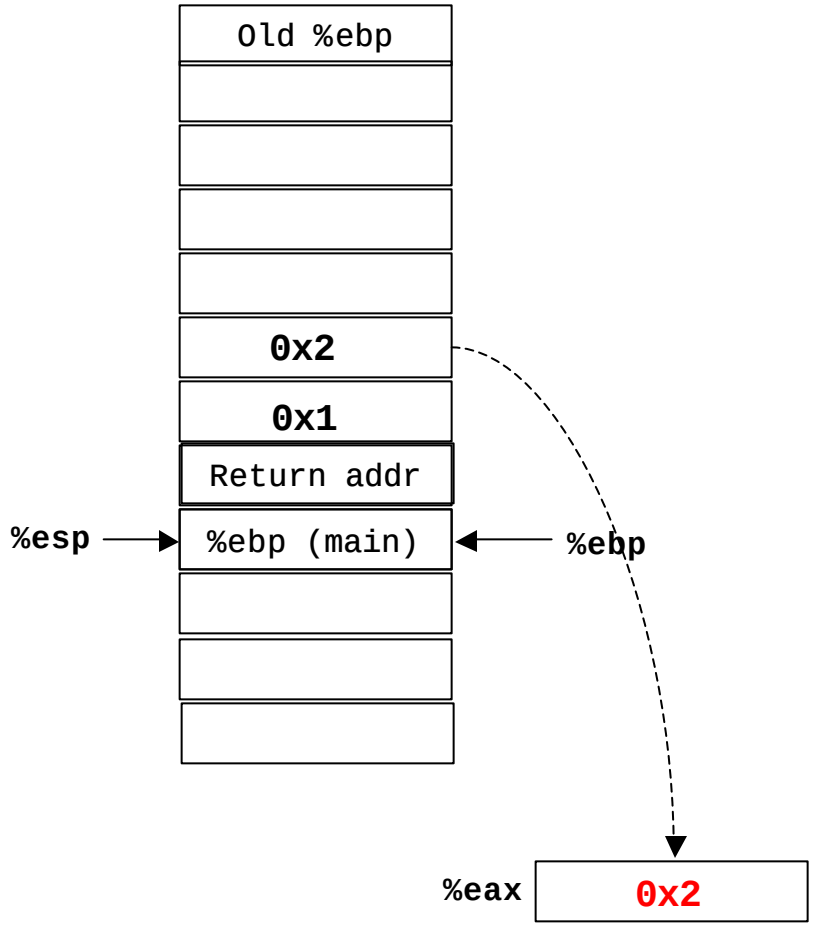
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

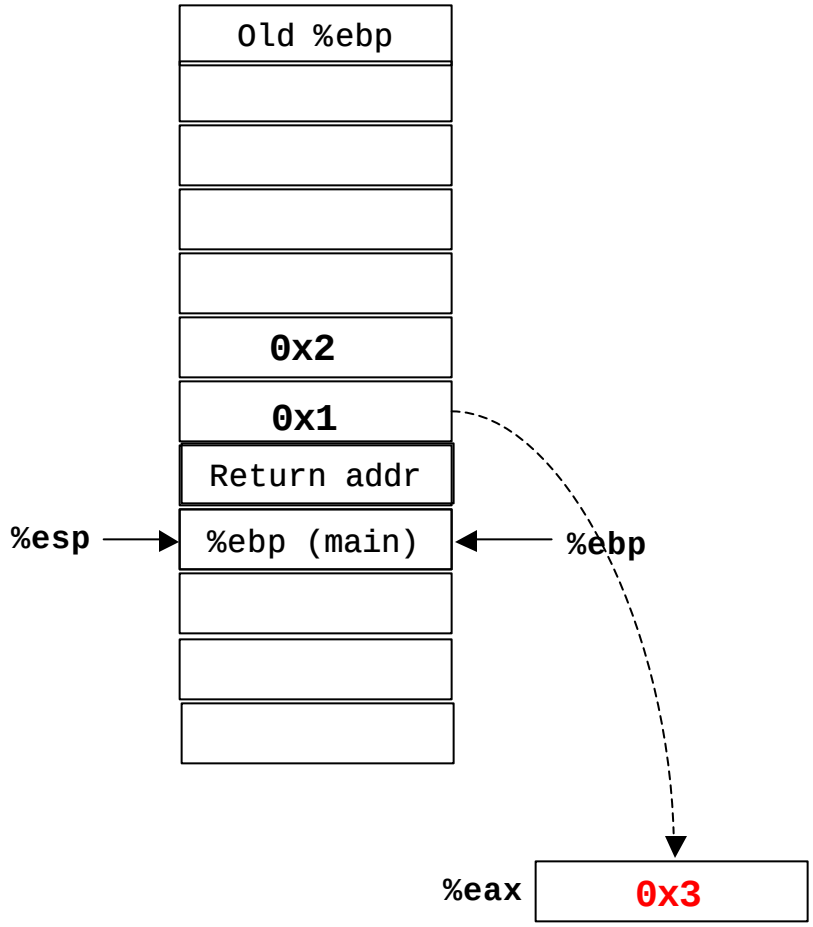
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

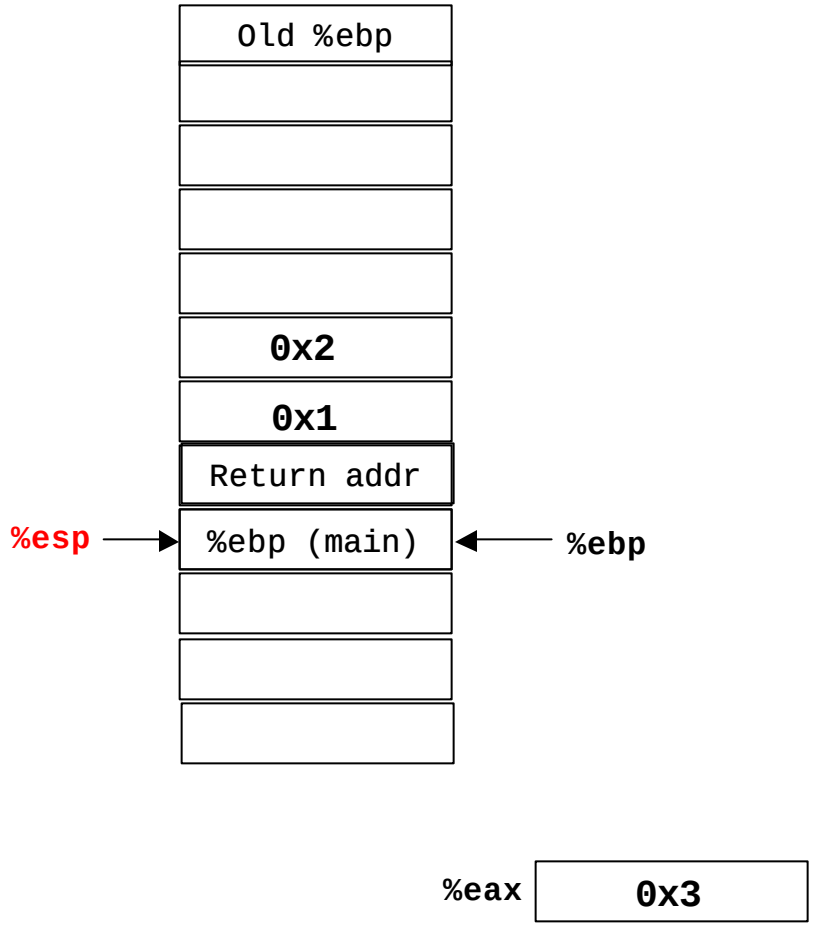
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

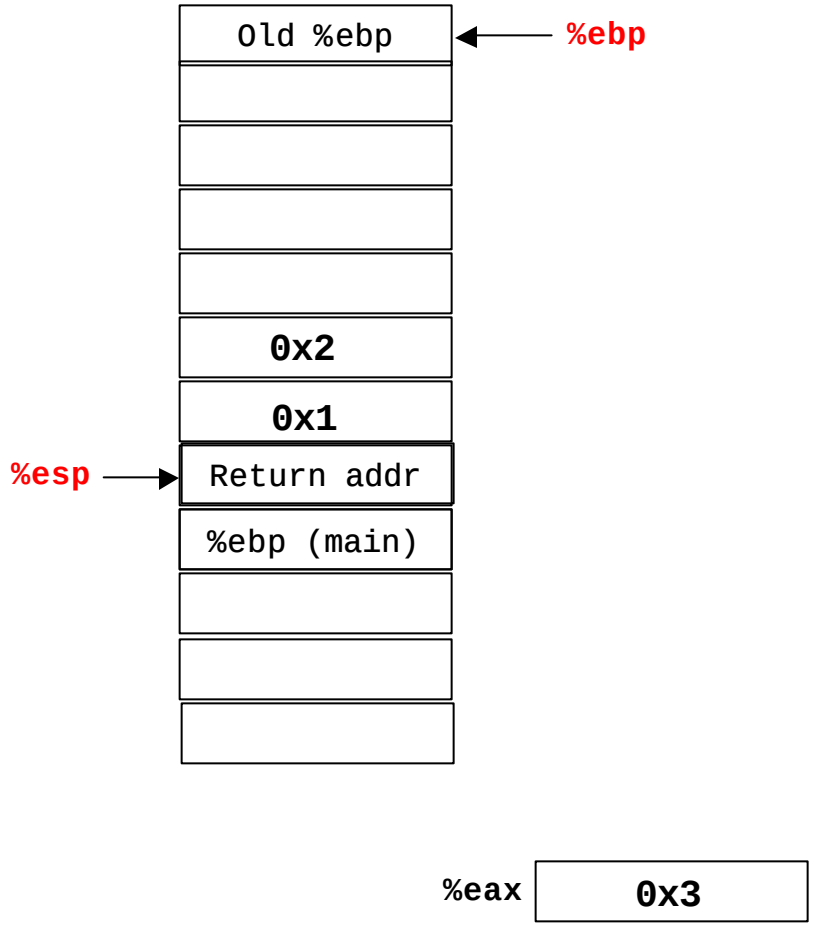
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

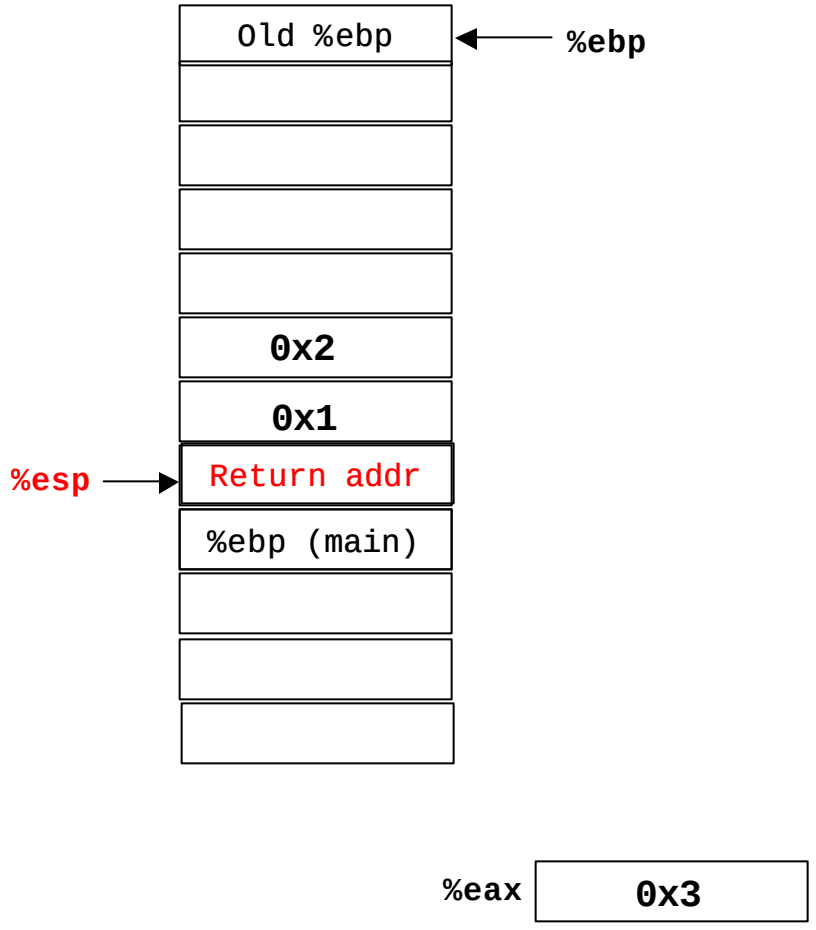
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

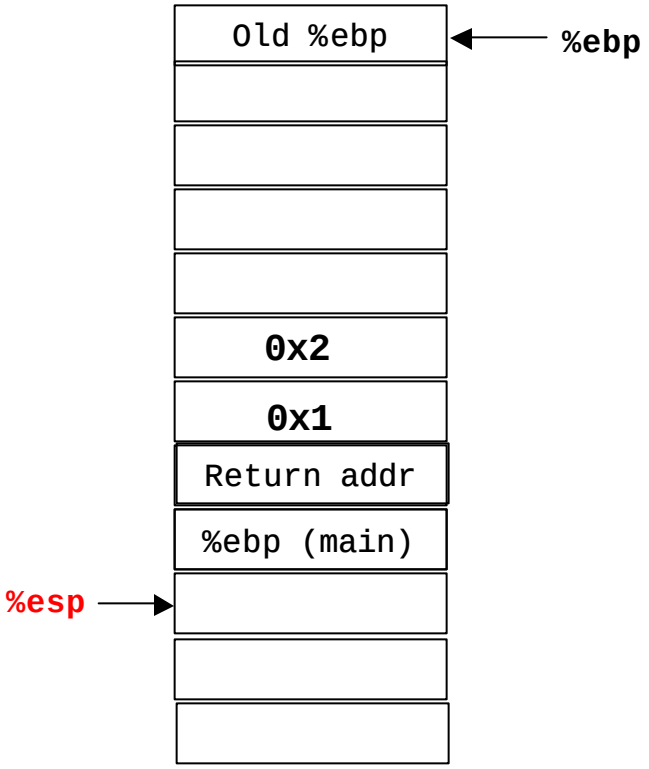
```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```

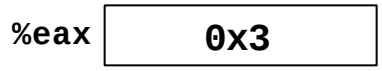
```
<example_1>:
        push   %ebp
        mov    %esp,%ebp
        mov    0xc(%ebp), %eax
        add    0x8(%ebp), %eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

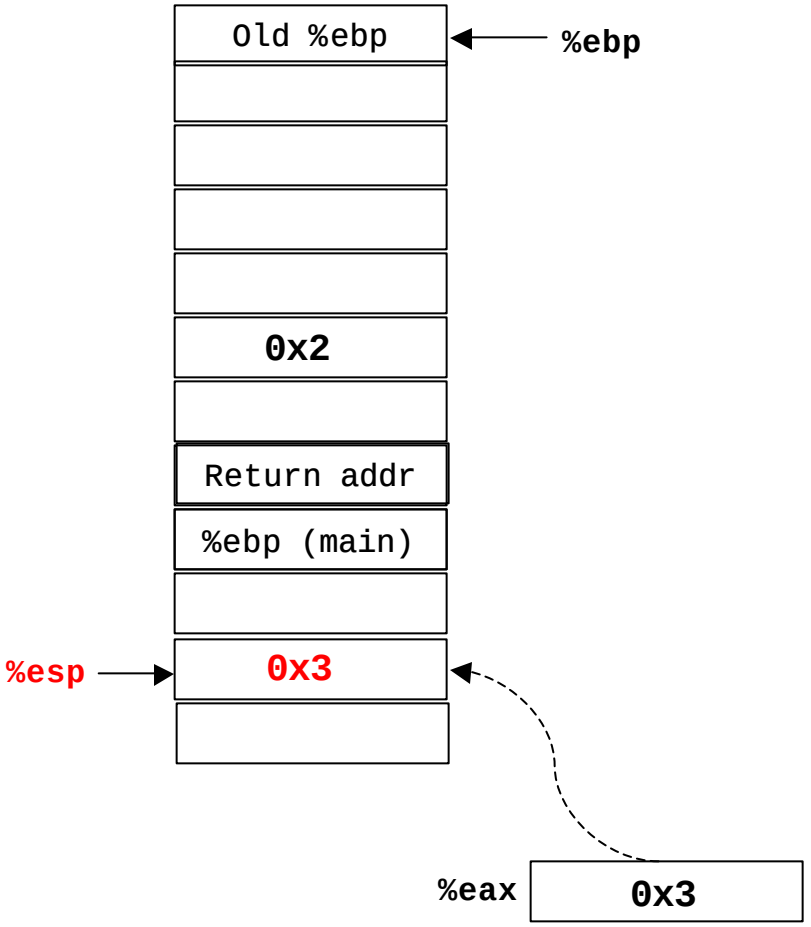
```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```





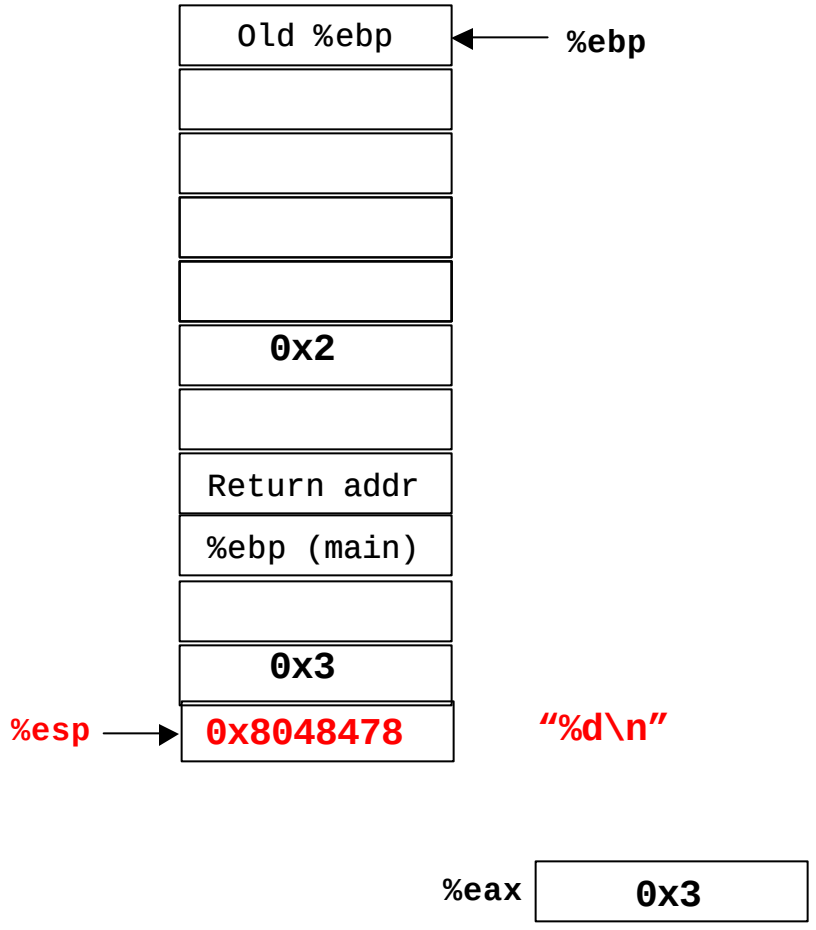
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



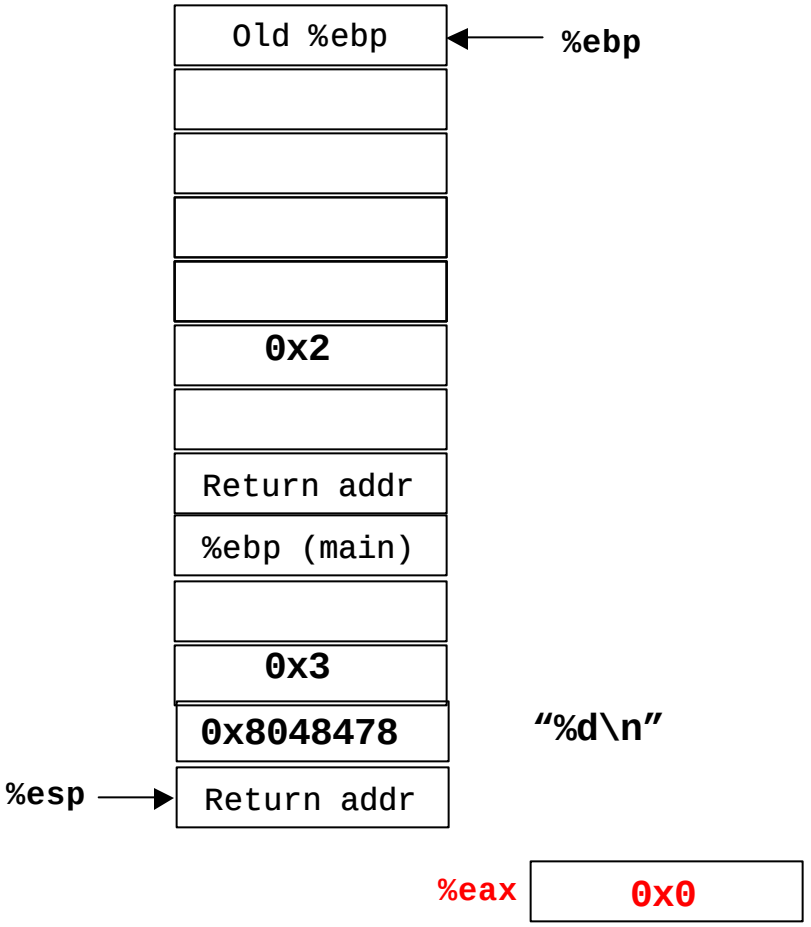
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



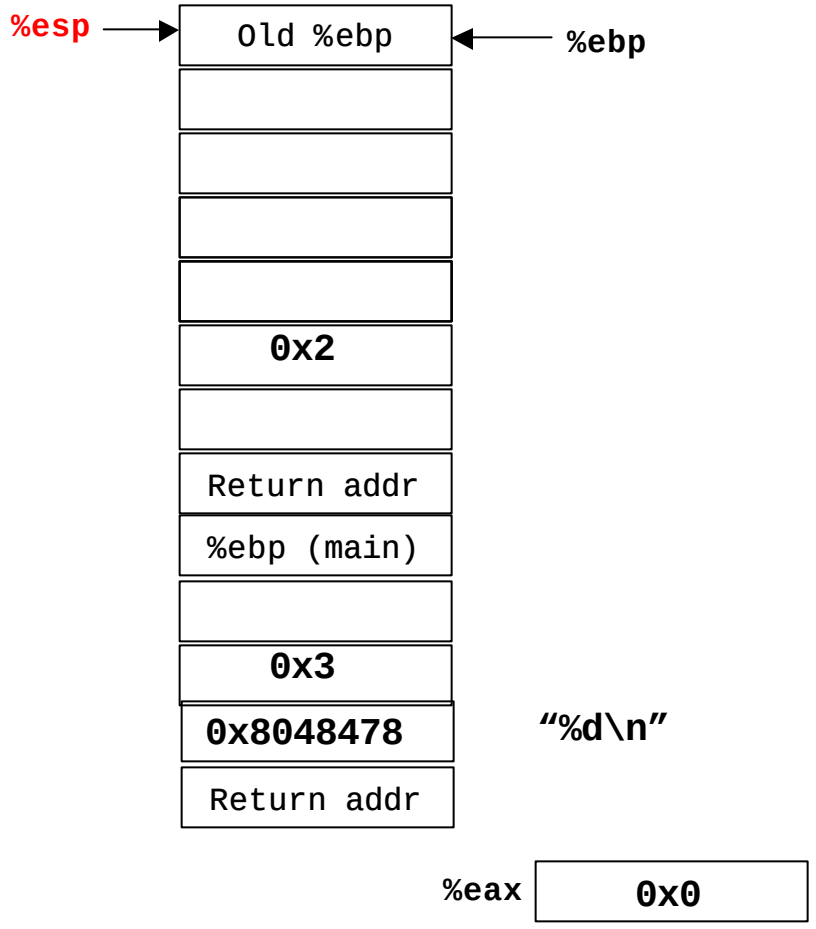
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



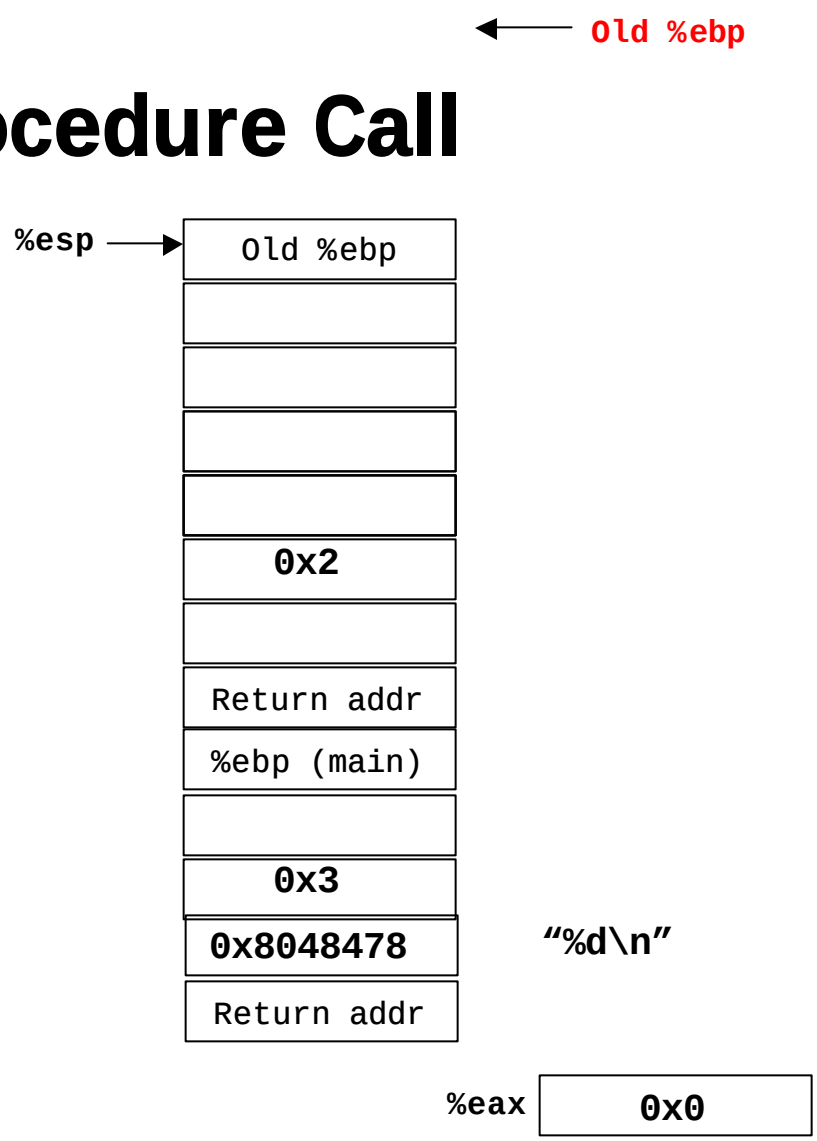
Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: Procedure Call

```
<main>:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        add    $0xfffffff8,%esp
        push   $0x2
        push   $0x1
        call   <example_1>
        add    $0xfffffff8,%esp
        push   %eax
        push   $0x8048478
        call   <printf>
        xor    %eax,%eax
        mov    %ebp,%esp
        pop    %ebp
        ret
```



Example 1: C code

```c
int example_1 (int x, int y)
{
  return x + y;
}


int main()
{
  int result;

  result = example_1(x,y);

  printf("%d\n", result);

  return 0;
}
```

Example 2: Recursion

Example 2: C code

```
int example_2 (int n)
{
  int result;
  if (n <= 2)
    result = 1;
  else
    result = example_2(n-2) + example_2(n-1);
}

int main()
{
  int n, result;
  printf(“n=“);
  scanf(“%d”, &n);
  result = example_2(n);
  printf(“example_2(%d) = %d\n", n, result);

  return 0;
}
```

Homogenous Data: Arrays

- Allocated as contiguous blocks of memory
- No boundary check in C!
- Address Computation Examples:

```
int cmu_class[5] = {213, 212, 122, 111, 451}
```

cmu_class begins at memory address 0x40

```
&cs_class[0]        40 + 4*0 = 40
&cs_class[3]        40 + 4*3 = 52
&cs_class[-1]       40 + 4*-1 = 36
&cs_class[15]       40 + 4*15 = 100
```

Example 3: Array

Example 3: C code

```
int example_3 (int x[], int num)
{
  int ii, sum;

  sum = 0;
  for (ii=0; ii<num; ii++)
    sum += x[ii];

  return sum;
}

int main()
{
  int x[5] = {1, 2, 3, 4, 5};
  printf("%d\n", example_3(x, 5));
  return 0;
}
```

Struct and Union

- Organize data
- **`struct`** stores multiple elements
- **`union`** stores one single element at a time
- Members of a union change how you look at data
- Unions are used for mutually exclusive data

```
struct one {
   int i;
   double d;
   char c[2];
}
```

```
union two {
   int i;
   double d;
   char c[2];
}
```

Example 4: Linked List

Example 4: C code