

Discussion 0

C, x86

01/20/23

Staff

Announcements

| Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|--------|-------------------|--------------------|-------------------|--------------------|---------------------|-----------------------------|
| | | Homework 0 Release | | | | |
| | Project 0 Release | | Homework 0 Due | Homework 1 Release | Early Drop Deadline | Group Formation Form Due |
| | | Project 0 Due | Project 1 Release | | | |

Important Policies

Contact staff through Ed.

- Use <u>cs162@eecs.berkeley.edu</u> to reach head TA and professors if Ed is insufficient.
- Only use individual emails for private matters.

3 midterms (check website for date and time).

• Exam conflict forms will be released ~3 weeks prior to exams.

Slip day policy:

- We provide 4 HW slip days and 5 Project slip days which are meant for emergencies
 - <u>Extension form</u> available for DSP and those with extenuating circumstances
 - Form is also accessible from the course website

Discussions 0 and 1 are optional.

- Feel free to attend different TAs' sections to find a teaching style that suits you best.
- Attendance for discussion 2 and onward are mandatory. Preference forms will be available soon.

Follow office hours policies of filling out a detailed ticket and being present in the OH room (Soda 347) when your ticket is taken.

- Almost all OH will be in person. There are a couple of online OHs meant specifically for those who absolutely cannot make it in person.
- Take advantage of empty office hours by starting assignments early.

Post your questions on Ed in the appropriate threads.

No private debugging posts allowed.

C

Types

C is **statically typed** (i.e. types are known at compile time).

C is **weakly typed** (i.e. can cast between any types).

Primitive types are char, short, int, long, float.

Arrays are contiguous pieces of memory of a homogenous type

- Denoted with [] (e.g. int [] for an array of integers).
- String is an array of chars with last element being null

Build compound types using structs

Also contiguous in memory

Pointers are references that hold the address of an object in memory.

- Essentially just unsigned integers.
- Prefix a pointer with * to return the value at the memory address that the pointer is holding.
- Prefix a variable with & to return the memory address of the variable.

| Little Endian | +3 | +2 | +1 | +0 | |
|---------------|-----|-----|-----|-----|---|
| 0x7FFFFFC | 00 | 00 | 00 | 08 | int a = 8 |
| 0x7FFFFFF8 | FF | FF | FF | FF | int b = −1 |
| 0x7FFFFFF4 | 7F | FF | FF | FC | int *p = &a |
| 0x7FFFFFF0 | 27 | CE | 00 | '0' | 27, CE are garbage |
| 0x7FFFFFEC | 'T' | 'N' | 'I' | 'P' | char s[] = "PINTO" |
| 0x7FFFFE8 | 00 | 00 | 00 | 02 | <pre>struct point { int x; int y;</pre> |
| 0x7FFFFFE4 | 00 | 00 | 00 | 10 | <pre>struct point pt = {16, 2}</pre> |

Memory

Typical C program is divided into five segments.

- **Text** contains machine code of the compiled program.
- (Un)initialized data contains (un)initialized global/static memory.
- **Heap** contains dynamically allocated memory.
- Stack contains local variables and arguments.
- Initialized strings and global constants may be stored in read-only segments (.rodata).

Think of memory as a giant array with elements of one byte.

• Memory addresses = indices of array.

Heap memory needs to be explicitly managed by the user.

- Allocate memory using malloc, calloc, realloc which return a pointer to a chunk of memory.
- Release memory using free.

High Address

| Command Line Arguments | | | |
|------------------------|--|--|--|
| Stack | | | |
| \ | | | |
| ↑ | | | |
| Неар | | | |
| Uninitialized Data | | | |
| Initialized Data | | | |
| Text | | | |

Low Address

GNU Debugger (GDB)

Need to learn how to use it for 162 even if you skidded by 61C without it.

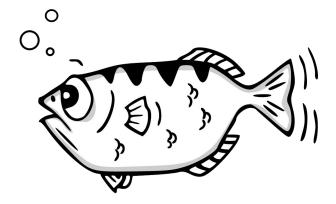
Staff will not help during OH unless you are able to use GDB.

General workflow of using GDB.

- 1. Compile program using –g flag.
- 2. Start GDB using gdb <executable name>.
- 3. Set breakpoints using break line number>. Can also break at functions using break <function name>.
- 4. Run program with run. If the program takes in arguments, pass in those after run (i.e. run arg1 arg2 ...).
- 5. Once breakpoint is hit, examine variables using print. Other commands like display, watch, and set are also useful.

Use GDB frequently to become familiar with the commands.

Check out GDB manual for more details.



1. Consider a valid double pointer char** dbl_char in a 32-bit system. What is returned by sizeof(*dbl_char)?

2. Consider strings char* a = "162 is the best" and char b[] = "162 is the best". Are a and b different?

3. Suppose you have an integer array int nums $[3] = \{152, 161, 162\}$. What are the differences between nums, &nums, and &nums [0]?

- Consider a valid double pointer char** dbl_char in a 32-bit system. What is returned by sizeof(*dbl_char)?
 Dereferencing a double pointer gives a single pointer. 32-bit systems have 32-bit = 4 byte memory addresses.
- 2. Consider strings char * a = "162 is the best" and char b[] = "162 is the best". Are a and b different?

3. Suppose you have an integer array int nums [3] = {152, 161, 162}. What are the differences between nums, &nums, and &nums [0]?

- Consider a valid double pointer char** dbl_char in a 32-bit system. What is returned by sizeof(*dbl_char)?
 Dereferencing a double pointer gives a single pointer. 32-bit systems have 32-bit = 4 byte memory addresses.
- 2. Consider strings char* a = "162 is the best" and char b[] = "162 is the best". Are a and b different?

 Yes. a points to a string literal in the read-only segment (.rodata) while b resides on the stack.
- 3. Suppose you have an integer array int $nums[3] = \{152, 161, 162\}$. What are the differences between nums, &nums, and &nums[0]?

- Consider a valid double pointer char** dbl_char in a 32-bit system. What is returned by sizeof(*dbl_char)?
 Dereferencing a double pointer gives a single pointer. 32-bit systems have 32-bit = 4 byte memory addresses.
- Consider strings char* a = "162 is the best" and char b[] = "162 is the best". Are a and b different?
 Yes. a points to a string literal in the read-only segment (.rodata) while b resides on the stack.
- 3. Suppose you have an integer array int nums[3] = {152, 161, 162}. What are the differences between nums, &nums, and &nums[0]?

 The three all return the same memory address, but nums and &nums[0] points to the first element of nums while &nums points to the entire list.

 Incrementing each pointer by 1 will result in increase of sizeof(int) for nums and &nums[0] but an increase of 3 * sizeof(int) for &nums.

```
#include <stdio.h>
                                                 typedef struct helper_args {
                                                                                                   #include "lib.h"
#include "lib.h"
                                                 #ifdef ABC
                                                                                                   char* helper_func(helper_args_t* args) {
                                                  char* aux:
int main(int argc, char** argv) {
                                                 #endif
                                                                                                     int i:
  helper_args_t helper_args;
                                                  char* string;
                                                                                                     for (i = 0; args->string[i] != '\0'; i++)
                                                                                                       if (args->string[i] == args->target)
  helper_args.string = argv[0];
                                                  char target;
  helper_args.target = '/';
                                                 } helper_args_t;
                                                                                                         return &args->string[i + 1];
  char* result = helper_func(&helper_args);
                                                 char* helper_func(helper_args_t* args);
                                                                                                     return args->string;
  printf("%s\n", result);
  return 0:
                      App.c
                                                                      lib.h
                                                                                                                       lib.c
```

You build the program on a 64-bit machine as follows.

```
> gcc -c app.c -o app.o
> gcc -c lib.c -o lib.o
> gcc app.o lib.o -o app
```

1. What is the size of a helper_args_t struct?

Suppose you add a #define ABC at the top of lib.h. What is the size of a helper_args_t struct?

```
#include <stdio.h>
#include "lib.h"

int main(int argc, char** argv) {
  helper_args_t helper_args;
  helper_args.string = argv[0];
  helper_args.target = '/';
  char* result = helper_func(&helper_args);
  printf("%s\n", result);
  return 0;
}
App.c
```

1. What is the size of a helper_args_t struct?

16 bytes. Only char* string and char target meaning 9 bytes but GCC pads structs.

```
typedef struct helper_args {
#ifdef ABC
  char* aux;
#endif
  char* string;
  char target;
} helper_args_t;
char* helper_func(helper_args_t* args);
```

lib.h

2. Suppose you add a #define ABC at the top of lib.h. What is the size of a helper_args_t struct?

```
#include "lib.h"

char* helper_func(helper_args_t* args) {
  int i;
  for (i = 0; args->string[i] != '\0'; i++)
    if (args->string[i] == args->target)
      return &args->string[i + 1];
  return args->string;
}
```

struct?

```
#include <stdio.h>
#include "lib.h"

int main(int argc, char** argv) {
  helper_args_t helper_args;
  helper_args.string = argv[0];
  helper_args.target = '/';
  char* result = helper_func(&helper_args);
  printf("%s\n", result);
  return 0;
}
App.c
```

1. What is the size of a helper_args_t

16 bytes. Only char* string and char target meaning 9 bytes but GCC pads structs.

```
typedef struct helper_args {
#ifdef ABC
  char* aux;
#endif
  char* string;
  char target;
} helper_args_t;
char* helper_func(helper_args_t* args);
```

lib.h

2. Suppose you add a #define ABC at the top of lib.h. What is the size of a helper_args_t struct?

24 bytes. Additional 8 bytes from char* aux since ABC is defined.

```
#include "lib.h"

char* helper_func(helper_args_t* args) {
   int i;
   for (i = 0; args->string[i] != '\0'; i++)
      if (args->string[i] == args->target)
        return &args->string[i + 1];
   return args->string;
}
```

```
#include <stdio.h>
#include "lib.h"

int main(int argc, char** argv) {
  helper_args_t helper_args;
  helper_args.string = argv[0];
  helper_args.target = '/';
  char* result = helper_func(&helper_args);
  printf("%s\n", result);
  return 0;
}
```

App.c

 Suppose you build the program in a different way with the original files (i.e. none of the changes from previous questions apply).

```
> gcc -DABC -c app.c -o app.o
> gcc -c lib.c -o lib.o
> gcc app.o lib.o -o app
```

The program will now exhibit undefined behavior. Why?

```
typedef struct helper_args {
#ifdef ABC
  char* aux;
#endif
  char* string;
  char target;
} helper_args_t;
char* helper_func(helper_args_t* args);
```

lib.h

```
#include "lib.h"

char* helper_func(helper_args_t* args) {
  int i;
  for (i = 0; args->string[i] != '\0'; i++)
    if (args->string[i] == args->target)
      return &args->string[i + 1];
  return args->string;
}
```

```
#include <stdio.h>
#include "lib.h"

int main(int argc, char** argv) {
  helper_args_t helper_args;
  helper_args.string = argv[0];
  helper_args.target = '/';
  char* result = helper_func(&helper_args);
  printf("%s\n", result);
  return 0;
}
```

App.c

 Suppose you build the program in a different way with the original files (i.e. none of the changes from previous questions apply).

```
> gcc -DABC -c app.c -o app.o
> gcc -c lib.c -o lib.o
> gcc app.o lib.o -o app
```

The program will now exhibit undefined behavior. Why?

```
typedef struct helper_args {
#ifdef ABC
  char* aux;
#endif
  char* string;
  char target;
} helper_args_t;
char* helper_func(helper_args_t* args);
```

lib.h

app.cis compiled with ABC defined but lib.cis not

- main stored argv[0] at address of helper_args +8
- helper_func access address of args when accessing args->string
- First 8 bytes of args which helper_func is accessing is garbage

```
#include "lib.h"

char* helper_func(helper_args_t* args) {
  int i;
  for (i = 0; args->string[i] != '\0'; i++)
    if (args->string[i] == args->target)
      return &args->string[i + 1];
  return args->string;
}
```

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
   sort(a, l, j-1);
   sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL;
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

.. We want to debug the program using GDB. How should we compile the program?

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
     ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
   sort(a, l, j-1);
   sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

 We want to debug the program using GDB. How should we compile the program?

gcc -g singer.c -o singer Need a -g flag for debugging.

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[i] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
   sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

2. When running the program without any arguments, what line does the segfault happen? Describe the memory operations happening in that line.

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++i:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break:
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted : " !Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

When running the program without any arguments, what line does the segfault happen? Describe the memory operations happening in that line.

Find segfaulting line by letting the program run until it encounters the fault.

```
> gcc -g singer.c -o singer
> gdb singer
(gdb) run
Starting program: /home/runner/intro/singer
Unsorted: "IU is the best char!"

Program received signal SIGSEGV, Segmentation
fault.
0x00005646308006c8 in swap (a=0x564630800904
"IU is the best singer!", i=1, j=21)
    at singer.c:6
```

Ignore "warning: Error disabling address space randomization: Operation not permitted" if using Replit.

a[i] = a[j];

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l]:
  int i = l, j = r+1;
  while (1) {
    do
      ++i:
    while (a[i] <= pivot && i <= r);</pre>
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
  if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
  if (argc > 1)
   a = argv[1];
  else
    a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted : " !Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

When running the program without any arguments, what line does the segfault happen? Describe the memory operations happening in that line.

Use backtrace for more comprehensive breakdown.

```
(gdb) backtrace
#0 0x00005646308006c8 in swap
(a=0x564630800904 "IU is the best singer!",
i=1, j=21)
    at singer.c:6
#1 0x0000564630800773 in partition
(a=0x564630800904 "IU is the best singer!",
l=0, r=21)
    at singer.c:26
#2 0x00005646308007bd in sort
(a=0x564630800904 "IU is the best singer!",
l=0, r=21)
    at singer.c:36
#3 0x0000564630800861 in main (argc=1,
argv=0x7ffd04ac7098) at singer.c:51
```

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
  if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
  if (argc > 1)
   a = argv[1];
  else
    a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

 When running the program without any arguments, what line does the segfault happen? Describe the memory operations happening in that line.

Two memory operations

- 1. Read from a [j].
- 2. Write to a [i].

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
   sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted : " !Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

3. Run the program with and without an argument and observe the memory addresses of a in the segfaulting line. Why are the memory addresses so different?

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted : " !Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

Run the program with and without an argument and observe the memory addresses of a in the segfaulting line. Why are the memory addresses so different?

Break at line 6 using GDB.

```
> gdb singer
(gdb) break 6
Breakpoint 1 at 0x6ab: file singer.c, line 6.
(gdb) run
Starting program: /home/runner/intro/singer
Unsorted: "IU is the best singer!"

Breakpoint 1, swap (
    a=0x5624e4600904 "IU is the best singer!",
i=1, j=21)
    at singer.c:6
6    a[i] = a[j];
(gdb) print a
$1 = 0x5624e4600904 "IU is the best singer!"
```

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++i:
    while (a[i] <= pivot && i <= r);</pre>
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break:
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted : " !Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

3. Run the program with and without an argument and observe the memory addresses of a in the segfaulting line. Why are the memory addresses so different?

Break at line 6 using GDB.

```
(gdb) run "Taeyeon is the best singer!"
The program being debugged has been started
already.
Start it from the beginning? (y or n) y
Starting program: /home/runner/intro/asuna
"Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"

Breakpoint 1, swap (
    a=0x7ffcce01bfe5 "Taeyeon is the best
singer!", i=1, j=26)
    at asuna.c:6
(gdb) print a
$2 = 0x7ffcce01bfe5 "Taeyeon is the best
singer!"
```

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++i:
    while (a[i] <= pivot && i <= r);</pre>
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeveon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

Run the program with and without an argument and observe the memory addresses of a in the segfaulting line. Why are the memory addresses so different?

No argument: 0x5624e4600904

 Statically defined strings stored in read-only segment (.rodata).

With argument: 0x7ffcce01bfe5

Arguments are passed in through the stack.

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
   sort(a, l, j-1);
   sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL;
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!";
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

How should the code be changed to fix the segfault?

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);</pre>
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
   a = "IU is the best singer!":
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

4. How should the code be changed to fix the segfault?

Write to a[i] is the problem since it's in read-only data \rightarrow need to put a in writable memory.

Allocate memory on the heap and put the default string on there.

- Accomplish with malloc followed by strcpy.
- Equivalently, use <u>strdup</u>.

Replace line 48.

```
void swap(char* a, int i, int j) {
  char t = a[i];
  a[i] = a[i];
  a[i] = t;
int partition(char* a, int l, int r){
  int pivot = a[l];
  int i = l, j = r+1;
  while (1) {
    do
      ++j:
    while (a[i] <= pivot && i <= r);
    do
      --i;
    while (a[j] > pivot);
    if (i >= j)
     break;
    swap(a, i, j);
  swap(a, l, j);
  return i:
```

```
void sort(char* a, int l, int r){
 if (l < r){
   int j = partition(a, l, r);
    sort(a, l, j-1);
    sort(a, j+1, r);
int main(int argc, char** argv){
  char* a = NULL:
 if (argc > 1)
   a = argv[1];
  else
    a = strdup("IU is the best singer!");
  printf("Unsorted: \"%s\"\n", a);
  sort(a, 0, strlen(a) - 1);
  printf("Sorted : \"%s\"\n", a);
> ./singer "Taeyeon is the best singer!"
Unsorted: "Taeyeon is the best singer!"
Sorted: "!Tabeeeeeghiinnorssstty"
> ./singer
Unsorted: "IU is the best singer!"
Segmentation fault (core dumped)
```

4. How should the code be changed to fix the segfault?

Write to a[i] is the problem since it's in read-only data \rightarrow need to put a in writable memory.

Allocate memory on the heap and put the default string on there.

- Accomplish with malloc followed by strcpy.
- Equivalently, use <u>strdup</u>.

Replace line 48.

x86

Registers

Registers are small storage spaces directly on the processor.

• Allows for fast memory access.

General purpose registers (GPR) store both data and addresses.

- x86 has 8, RISC-V has 32 (x0-x31).
- Started as 16-bits, extend to 32-bit using e prefix (e.g. EAX for AX).
- Access 8-bit LSB by replacing last letter with I (e.g. AL for AX).
- Access 8-bit MSB by replacing last letter with h (e.g. AH for AX) only for AX, BX, CX, DX.

Instruction pointer register holds address of next instruction to execute.

- Called ip which can be extended with e prefix like a GPR.
- Can't be read/modified like a GPR using regular memory instructions.

| | Name | Purpose |
|----|-------------------|---|
| AX | Accumulator | I/O port access, arithmetic, interrupt calls |
| ВХ | Base | Base pointer for memory access |
| CX | Counter | Loop counting, bit shifts |
| DX | Data | I/O port access, arithmetic, interrupt calls |
| SP | Stack Pointer | Top address of stack |
| BP | Base Pointer | Base address of stack |
| SI | Source Index | Source for stream operations (e.g. string) |
| DI | Destination Index | Destination for stream operations (e.g. string) |

Syntax

Use AT&T Syntax not Intel which is used by GCC.

Prefix registers with % (e.g. %eax), constants with \$ (e.g. \$4).

General structure is **inst src, dest**.

Address memory with offset(base, index, scale).

- base, index = registers, offset = any integer, scale = 1, 2, 4, or 8.
- Accesses data at address base + index * scale + offset.
- All parameters optional but will see base and offset usually.
- Using lea instruction will operate on the address itself not contents.

Suffix instructions to signify operand size.

• Not always necessary but should use them regardless.

| mov 8(%ebx), %eax | Move contents from address EBX + 8 into EAX |
|-----------------------------|--|
| mov %ecx, -4(%esi, %ebx, 8) | Move contents in ECX into address ESI + 8 * EBX - 4 |
| lea 8(%ebx), %eax | Puts EBX + 8 into EAX |

| movb \$0, (%esp) | Zero out a single byte from ESP |
|------------------|---------------------------------|
| movw \$0, (%esp) | Zero out two bytes from ESP |
| movl \$0, (%esp) | Zero out four bytes from ESP |

Calling convention is a procedure for how to call and return from functions.

- Specifies stack management, argument passing, register saving, etc.
- One set of rule each for the caller and callee

Many different calling conventions, will use the one defined in i386 System V ABI in this class.

EBP Existing Caller Stack Frame ESP

Caller

Before calling the function (i.e. prologue),

1. Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.

EBP

Existing Caller Stack Frame

Caller-Saved GPRs

ESP

Caller

Before calling the function (i.e. prologue),

- Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.
- Push parameters onto the stack in reverse order. Add necessary padding before the parameters to ensure a 16-byte alignment.

EBP Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters ESP**

Caller

Before calling the function (i.e. prologue),

- 1. Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.
- 2. Push parameters onto the stack in reverse order. Add necessary padding *before the parameters* to ensure a 16-byte alignment.

Call function by pushing the return address onto the stack and jumping to the function.

EBP Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters** Return Address **ESP**

Callee

Before executing any function logic (i.e. prologue),

1. Push EBP onto the stack and set EBP to be the new ESP.

Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters** Return Address **EBP of Caller Stack Frame**

EBP, ESP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.

| Existing Caller Stack Frame |
|-----------------------------|
| Caller-Saved GPRs |
| Padding |
| Function Parameters |
| Return Address |
| EBP of Caller Stack Frame |
| Local Variables |
| |
| |

EBP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

| Existing Caller Stack Frame |
|-----------------------------|
| Caller-Saved GPRs |
| Padding |
| Function Parameters |
| Return Address |
| EBP of Caller Stack Frame |
| Local Variables |
| Callee-Saved GPRs |

EBP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

| Existing Caller Stack Frame |
|-----------------------------|
| Caller-Saved GPRs |
| Padding |
| Function Parameters |
| Return Address |
| EBP of Caller Stack Frame |
| Local Variables |
| Callee-Saved GPRs |

EBP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

1. Store return value in EAX.

| Existing Caller Stack Frame |
|-----------------------------|
| Caller-Saved GPRs |
| Padding |
| Function Parameters |
| Return Address |
| EBP of Caller Stack Frame |
| Local Variables |
| Callee-Saved GPRs |

EBP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

- 1. Store return value in EAX.
- 2. Restore callee-saved GPRs if any from the prologue.

| Existing Caller Stack Frame |
|-----------------------------|
| Caller-Saved GPRs |
| Padding |
| Function Parameters |
| Return Address |
| EBP of Caller Stack Frame |
| Local Variables |
| Callee Saved GPRs |

EBP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

- 1. Store return value in FAX.
- 2. Restore callee-saved GPRs if any from the prologue.
- 3. Deallocate local variables.

Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters** Return Address FBP of Caller Stack Frame

EBP, ESP

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

- 1. Store return value in EAX.
- 2. Restore callee-saved GPRs if any from the prologue.
- Deallocate local variables.
- 4. Restore caller's EBP from stack.

EBP

Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters** Return Address FBP of Caller Stack Frame

Callee

Before executing any function logic (i.e. prologue),

- 1. Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

- 1. Store return value in EAX.
- 2. Restore callee-saved GPRs if any from the prologue.
- Deallocate local variables.
- 4. Restore caller's EBP from stack.
- Return from function call by popping the return address pushed by the caller in its prologue and jumping to it.

EBP

FSP

Existing Caller Stack Frame Caller-Saved GPRs **Padding Function Parameters** Return Address

Caller

Before calling the function (i.e. prologue),

- 1. Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.
- 2. Push parameters onto the stack in reverse order. Add necessary padding before the parameters to ensure a 16-byte alignment.

Call function by pushing the return address onto the stack and jumping to the function.

Once function call returns (i.e. epilogue),

1. Remove parameters from the stack.

EBP

ESP

Existing Caller Stack Frame Caller-Saved GPRs

Caller

Before calling the function (i.e. prologue),

- 1. Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.
- 2. Push parameters onto the stack in reverse order. Add necessary padding *before the parameters* to ensure a 16-byte alignment.

Call function by pushing the return address onto the stack and jumping to the function.

Once function call returns (i.e. epilogue),

- 1. Remove parameters from the stack.
- 2. Restore caller-saved GPRs if any from the prologue.

EBP

ESP

Existing Caller Stack Frame

Caller

Before calling the function (i.e. **prologue**),

- 1. Save caller-saved GPRs (EAX, ECX, EDX) onto the stack if needed after the function call.
- 2. Push parameters onto the stack in reverse order. Add necessary padding before the parameters to ensure a 16-byte alignment.

Call function by pushing the return address onto the stack and jumping to the function.

Once function call returns (i.e. epilogue),

- 1. Remove parameters from the stack.
- 2. Restore caller-saved GPRs if any from the prologue.

Callee

Before executing any function logic (i.e. prologue),

- Push EBP onto the stack and set EBP to be the new ESP.
- 2. Allocate stack space for local variables.
- 3. Save callee-saved GPRs (EBX, EDI, ESI) onto the stack if used during the function logic.

Perform function logic.

Before returning (i.e. epilogue),

- Store return value in EAX.
- 2. Restore callee-saved GPRs if any from the prologue.
- 3. Deallocate local variables.
- 4. Restore caller's EBP from stack.
- Return from function call by popping the return address pushed by the caller in its prologue and jumping to it.

| Instruction | Purpose | Effective |
|-------------|---|-------------------------------------|
| pushl src | Push src onto stack | subl \$4, %esp movl src, (%esp) |
| popl dest | Pop from stack into dest | movl (%esp), dest addl \$4, %esp |
| call addr | Push return address onto stack and jump to addr | pushl %eip jump addr |
| leave | Restore EBP and ESP to previous stack frame | movl %ebp, %esp popl %ebp |
| ret | Pop return address from stack and jump to it | popl %eip |

| 1. | Between SP and BP, which has a higher memory address? |
|----|---|
| | |

2. Based on the differences of RISC and CISC, why might x86 have fewer GPRs compared to RISC-V?

3. Write three different ways to clear the \code{eax} register (i.e. store a 0).

4. True or False: Right before the caller jumps to the desired function, the stack must be 16-byte aligned.

- Between SP and BP, which has a higher memory address?
 BP. Stack grows downwards → top of the stack (SP) moves towards lower addresses
- 2. Based on the differences of RISC and CISC, why might x86 have fewer GPRs compared to RISC-V?

3. Write three different ways to clear the EAX register (i.e. store a 0).

4. True or False: Right before the caller jumps to the desired function, the stack must be 16-byte aligned.

- Between SP and BP, which has a higher memory address?
 BP. Stack grows downwards → top of the stack (SP) moves towards lower addresses
- Based on the differences of RISC and CISC, why might x86 have fewer GPRs compared to RISC-V?
 Reduced instruction set → less hardware space needed for transistors → more room for GPRs
- 3. Write three different ways to clear the EAX register (i.e. store a 0).

4. True or False: Right before the caller jumps to the desired function, the stack must be 16-byte aligned.

- Between SP and BP, which has a higher memory address?
 BP. Stack grows downwards → top of the stack (SP) moves towards lower addresses
- Based on the differences of RISC and CISC, why might x86 have fewer GPRs compared to RISC-V?
 Reduced instruction set → less hardware space needed for transistors → more room for GPRs
- Write three different ways to clear the EAX register (i.e. store a 0).movl \$0, %eax subl %eax, %eax xorl %eax, %eax
- 4. True or False: Right before the caller jumps to the desired function, the stack must be 16-byte aligned.

Between SP and BP, which has a higher memory address?
 BP. Stack grows downwards → top of the stack (SP) moves towards lower addresses

Based on the differences of RISC and CISC, why might x86 have fewer GPRs compared to RISC-V?
 Reduced instruction set → less hardware space needed for transistors → more room for GPRs

3. Write three different ways to clear the EAX register (i.e. store a 0).

movl \$0, %eax subl %eax, %eax xorl %eax, %eax

4. True or False: Right before the caller jumps to the desired function, the stack must be 16-byte aligned.

False. Stack needs to be 16-byte aligned after parameters have been pushed onto the stack. Return address is pushed right before jumping.

```
%eax, -24(%ebp)
int main(void) {
                                       1 main:
                                                                              28
                                                                                        cmpl
  for (int a = 1; _____) {
                                       2
                                                pushl
                                                        %ebp
                                                                              29
                                                                                        jne
                                                                                                 .L4
                                                                                                -4(%ebp), %eax
                                                movl
                                                        %esp, %ebp
                                                                              30
                                                                                        movl
    int a2 = ____;
   for (int b = _____) {
                                       4
                                                subl
                                                        $32, %esp
                                                                              31
                                                                                        imull
                                                                                                -8(%ebp), %eax
                                                        $1, -4(%ebp)
      int b2 = ____;
                                                movl
                                                                              32
                                                                                        imull
                                                                                                 -20(%ebp), %eax
      int c = ____;
                                                jmp
                                                        .L2
                                                                              33
                                                                                        jmp
                                                                                                 .L5
     int c2 = ____;
                                       7 .17:
                                                                              34 .L4:
      if (_____) {
                                                movl
                                                        -4(%ebp), %eax
                                                                              35
                                                                                         addl
                                                                                                $1, -8(%ebp)
                                       8
        return ____;
                                                imull
                                       9
                                                        %eax, %eax
                                                                              36 .L3:
                                       10
                                                movl
                                                        %eax, -12(%ebp)
                                                                              37
                                                                                                $666, -8(%ebp)
                                                                                        cmpl
                                                        -4(%ebp), %eax
                                       11
                                                movl
                                                                              38
                                                                                        jle
                                                                                                 .L6
                                       12
                                                movl
                                                        %eax, -8(%ebp)
                                                                              39
                                                                                        addl
                                                                                                $1, -4(%ebp)
  return 0;
                                       13
                                                        .L3
                                                                              40 .L2:
                                                jmp
                                                                                                $333, -4(%ebp)
                                       14 .L6:
                                                                              41
                                                                                        cmpl
                                       15
                                                movl
                                                        -8(%ebp), %eax
                                                                              42
                                                                                        jle
                                                                                                 .L7
                                       16
                                                imull
                                                        %eax, %eax
                                                                              43
                                                                                        movl
                                                                                                $0, %eax
      What is the memory address of a relative
                                       17
                                                movl
                                                        %eax, -16(%ebp)
                                                                              44 .L5:
      to the base pointer?
                                                                                        leave
                                       18
                                                movl
                                                        $1000, %eax
                                                                              45
                                       19
                                                subl
                                                        -4(%ebp), %eax
                                                                              46
                                                                                         ret
                                       20
                                                subl
                                                        -8(%ebp), %eax
                                       21
                                                movl
                                                        %eax, -20(%ebp)
                                       22
                                                movl
                                                        -20(%ebp), %eax
                                       23
                                                imull
                                                        %eax, %eax
                                       24
                                                movl
                                                        %eax, -24(%ebp)
```

25

26

27

movl

movl

addl

-12(%ebp), %edx

-16(%ebp), %eax

%edx, %eax

```
int main(void) {
                                        1 main:
                                                                                28
                                                                                          cmpl
                                                                                                  %eax, -24(%ebp)
  for (int a = 1; _____) {
                                        2
                                                 pushl
                                                         %ebp
                                                                                29
                                                                                          jne
                                                                                                   .L4
                                                                                                  -4(%ebp), %eax
                                                 movl
                                                         %esp, %ebp
                                                                                30
                                                                                          movl
    int a2 = ____;
   for (int b = _____) {
                                        4
                                                 subl
                                                         $32, %esp
                                                                                31
                                                                                          imull
                                                                                                  -8(%ebp), %eax
      int b2 = ____;
                                                 movl
                                                         $1, -4(%ebp)
                                                                                32
                                                                                          imull
                                                                                                  -20(%ebp), %eax
      int c = ____;
                                                 jmp
                                                         .L2
                                                                                33
                                                                                          jmp
                                                                                                   .L5
      int c2 = ____;
                                        7 .17:
                                                                                34 .L4:
      if (_____) {
                                                 movl
                                                         -4(%ebp), %eax
                                                                                35
                                                                                          addl
                                                                                                  $1, -8(%ebp)
                                        8
        return ____;
                                                 imull
                                        9
                                                         %eax, %eax
                                                                                36 .L3:
                                        10
                                                 movl
                                                         %eax, -12(%ebp)
                                                                                37
                                                                                                  $666, -8(%ebp)
                                                                                          cmpl
                                                         -4(%ebp), %eax
                                        11
                                                 movl
                                                                                38
                                                                                          jle
                                                                                                  .L6
                                        12
                                                 movl
                                                         %eax, -8(%ebp)
                                                                                39
                                                                                          addl
                                                                                                  $1, -4(%ebp)
  return 0;
                                        13
                                                         .L3
                                                                                40 .L2:
                                                 jmp
                                                                                                  $333, -4(%ebp)
                                        14 .L6:
                                                                                41
                                                                                          cmpl
                                        15
                                                 movl
                                                         -8(%ebp), %eax
                                                                                42
                                                                                          jle
                                                                                                   .L7
                                                 imull
                                                         %eax, %eax
                                        16
                                                                                43
                                                                                          movl
                                                                                                  $0, %eax
      What is the memory address of a relative
                                        17
                                                 movl
                                                         %eax, -16(%ebp)
                                                                                44 .L5:
      to the base pointer?
                                                                                          leave
                                        18
                                                 movl
                                                         $1000, %eax
                                                                                45
                                        19
                                                 subl
                                                         -4(%ebp), %eax
                                                                                46
                                                                                          ret
      4 bytes below the base pointer (i.e. EBP -
                                                         -8(%ebp), %eax
                                        20
                                                 subl
      4).
                                        21
                                                 movl
                                                         %eax, -20(%ebp)
                                        22
                                                 movl
                                                         -20(%ebp), %eax
                                        23
                                                 imull
                                                         %eax, %eax
                                                         %eax, -24(%ebp)
                                        24
                                                 movl
                                        25
                                                 movl
                                                         -12(%ebp), %edx
                                        26
                                                 movl
                                                         -16(%ebp), %eax
                                        27
                                                 addl
                                                         %edx, %eax
```

```
%eax, -24(%ebp)
int main(void) {
                                       1 main:
                                                                               28
                                                                                         cmpl
  for (int a = 1; _____) {
                                       2
                                                pushl
                                                        %ebp
                                                                               29
                                                                                         jne
                                                                                                 .L4
                                                                                                 -4(%ebp), %eax
                                                movl
                                                        %esp, %ebp
                                                                               30
                                                                                         movl
    int a2 = ____;
   for (int b = _____) {
                                       4
                                                subl
                                                        $32, %esp
                                                                               31
                                                                                         imull
                                                                                                 -8(%ebp), %eax
                                                        $1, -4(%ebp)
      int b2 = ____;
                                                movl
                                                                               32
                                                                                         imull
                                                                                                 -20(%ebp), %eax
     int c = ____;
                                                                               33
                                                jmp
                                                         .L2
                                                                                         jmp
                                                                                                 .L5
     int c2 = ____;
                                       7 .17:
                                                                               34 .L4:
      if (_____) {
                                                movl
                                                        -4(%ebp), %eax
                                                                               35
                                                                                         addl
                                                                                                 $1, -8(%ebp)
                                       8
        return ____;
                                                imull
                                       9
                                                        %eax, %eax
                                                                               36 .L3:
                                       10
                                                movl
                                                        %eax, -12(%ebp)
                                                                               37
                                                                                                 $666, -8(%ebp)
                                                                                         cmpl
                                                        -4(%ebp), %eax
                                       11
                                                movl
                                                                               38
                                                                                         jle
                                                                                                 .L6
                                       12
                                                movl
                                                        %eax, -8(%ebp)
                                                                               39
                                                                                         addl
                                                                                                 $1, -4(%ebp)
  return 0:
                                       13
                                                         .L3
                                                                               40 .L2:
                                                jmp
                                                                                                 $333, -4(%ebp)
                                       14 .L6:
                                                                               41
                                                                                         cmpl
                                       15
                                                movl
                                                        -8(%ebp), %eax
                                                                               42
                                                                                         jle
                                                                                                 .L7
                                       16
                                                imull
                                                        %eax, %eax
                                                                               43
                                                                                         movl
                                                                                                 $0, %eax
      What is the end condition for the outer
                                       17
                                                movl
                                                        %eax, -16(%ebp)
                                                                               44 .L5:
      loop using a?
                                       18
                                                movl
                                                        $1000, %eax
                                                                               45
                                                                                         leave
                                       19
                                                subl
                                                        -4(%ebp), %eax
                                                                               46
                                                                                         ret
                                       20
                                                subl
                                                        -8(%ebp), %eax
                                       21
                                                movl
                                                        %eax, -20(%ebp)
                                       22
                                                movl
                                                        -20(%ebp), %eax
                                       23
                                                imull
                                                        %eax, %eax
                                       24
                                                movl
                                                        %eax, -24(%ebp)
                                       25
                                                movl
                                                        -12(%ebp), %edx
                                       26
                                                movl
                                                        -16(%ebp), %eax
                                       27
                                                addl
                                                        %edx, %eax
```

```
int main(void) {
                                    1 main:
 for (int a = 1; _____) {
                                    2
                                             pushl
                                             movl
   int a2 = ____;
   for (int b = _____) {
                                    4
                                             subl
     int b2 = ____;
                                             movl
     int c = ____;
                                             jmp
     int c2 = ____;
                                    7 .17:
     if (_____) {
                                             movl
                                    8
       return ____;
                                             imull
                                    9
                                    10
                                             movl
                                    11
                                             movl
                                    12
                                             movl
 return 0;
                                    13
                                             jmp
                                    14 .L6:
                                    15
                                             movl
                                    16
                                             imull
     What is the end condition for the outer
```

What is the end condition for the outer loop using a?
 a must be greater than 333.

```
%ebp
                 %esp, %ebp
                 $32, %esp
                 $1, -4(%ebp)
                  .L2
                 -4(%ebp), %eax
                 %eax, %eax
                 %eax, -12(%ebp)
                 -4(%ebp), %eax
                 %eax, -8(%ebp)
                 .L3
                 -8(%ebp), %eax
                 %eax, %eax
17
         movl
                 %eax, -16(%ebp)
18
         movl
                 $1000, %eax
19
         subl
                 -4(%ebp), %eax
                 -8(%ebp), %eax
20
         subl
21
         movl
                 %eax, -20(%ebp)
22
         movl
                 -20(%ebp), %eax
23
         imull
                 %eax, %eax
                 %eax, -24(%ebp)
24
         movl
25
         movl
                 -12(%ebp), %edx
26
         movl
                 -16(%ebp), %eax
27
         addl
                 %edx, %eax
```

```
28
          cmpl
                  %eax, -24(%ebp)
29
          jne
                   .L4
                  -4(%ebp), %eax
30
          movl
31
          imull
                  -8(%ebp), %eax
32
          imull
                  -20(%ebp), %eax
33
          jmp
                   .L5
34 .L4:
35
          addl
                  $1, -8(%ebp)
36 .L3:
37
                  $666, -8(%ebp)
          cmpl
38
          jle
                   .L6
39
          addl
                  $1, -4(%ebp)
40 .L2:
                  $333, -4(%ebp)
41
          cmpl
42
          jle
                   .L7
43
          movl
                  $0, %eax
44 .L5:
45
          leave
46
          ret
```

```
int main(void) {
  for (int a = 1; ______) {
    int a2 = ____;
    for (int b = ______) {
       int b2 = ____;
       int c = ____;
       int c2 = ____;
       if (______) {
         return ____;
       }
    }
   return 0;
}
```

3. What are the memory addresses the local variables (a2, b, b2, c, c2) relative to the base pointer?

```
1 main:
2
         pushl
                 %ebp
         movl
                 %esp, %ebp
4
         subl
                 $32, %esp
                 $1, -4(%ebp)
         movl
         jmp
                  .L2
7 .17:
         movl
                 -4(%ebp), %eax
8
         imull
9
                 %eax, %eax
10
         movl
                 %eax, -12(%ebp)
                 -4(%ebp), %eax
11
         movl
12
         movl
                 %eax, -8(%ebp)
13
                  .L3
         jmp
14 .L6:
15
         movl
                 -8(%ebp), %eax
16
         imull
                 %eax, %eax
17
         movl
                 %eax, -16(%ebp)
18
         movl
                 $1000, %eax
19
         subl
                 -4(%ebp), %eax
                 -8(%ebp), %eax
20
         subl
21
         movl
                 %eax, -20(%ebp)
22
         movl
                 -20(%ebp), %eax
23
         imull
                 %eax, %eax
                 %eax, -24(%ebp)
24
         movl
25
         movl
                 -12(%ebp), %edx
26
         movl
                 -16(%ebp), %eax
27
         addl
                 %edx, %eax
```

```
%eax, -24(%ebp)
28
          cmpl
29
          jne
                   .L4
                  -4(%ebp), %eax
30
          movl
31
          imull
                  -8(%ebp), %eax
32
          imull
                  -20(%ebp), %eax
33
          jmp
                   .L5
34 .L4:
35
          addl
                  $1, -8(%ebp)
36 .L3:
37
                  $666, -8(%ebp)
          cmpl
38
          jle
                   .L6
39
          addl
                  $1, -4(%ebp)
40 .L2:
                  $333, -4(%ebp)
41
          cmpl
42
          jle
                   .L7
43
          movl
                  $0, %eax
44 .L5:
          leave
45
46
          ret
```

```
int main(void) {
                                        1 main:
                                                                                 28
                                                                                           cmpl
                                                                                                   %eax, -24(%ebp)
  for (int a = 1; _____) {
                                        2
                                                 pushl
                                                          %ebp
                                                                                 29
                                                                                           jne
                                                                                                    .L4
                                                                                                   -4(%ebp), %eax
                                                 movl
                                                          %esp, %ebp
                                                                                 30
                                                                                           movl
    int a2 = ____;
    for (int b = _____) {
                                        4
                                                  subl
                                                          $32, %esp
                                                                                 31
                                                                                           imull
                                                                                                   -8(%ebp), %eax
                                                          $1, -4(%ebp)
      int b2 = ____;
                                                 movl
                                                                                 32
                                                                                           imull
                                                                                                   -20(%ebp), %eax
      int c = ____;
                                                 jmp
                                                          .L2
                                                                                 33
                                                                                           jmp
                                                                                                    .L5
      int c2 = ____;
                                        7 .17:
                                                                                 34 .L4:
      if (_____) {
                                                 movl
                                                          -4(%ebp), %eax
                                                                                 35
                                                                                           addl
                                                                                                   $1, -8(%ebp)
                                        8
        return ____;
                                                 imull
                                        9
                                                          %eax, %eax
                                                                                 36 .L3:
                                        10
                                                 movl
                                                          %eax, -12(%ebp)
                                                                                 37
                                                                                                   $666, -8(%ebp)
                                                                                           cmpl
                                                          -4(%ebp), %eax
                                        11
                                                 movl
                                                                                 38
                                                                                           jle
                                                                                                    .L6
                                        12
                                                          %eax, -8(%ebp)
                                                                                 39
                                                                                           addl
                                                                                                   $1, -4(%ebp)
                                                 movl
  return 0;
                                        13
                                                          .L3
                                                                                 40 .L2:
                                                 jmp
                                                                                                   $333, -4(%ebp)
                                        14 .L6:
                                                                                 41
                                                                                           cmpl
                                        15
                                                 movl
                                                          -8(%ebp), %eax
                                                                                 42
                                                                                           jle
                                                                                                    .L7
                                        16
                                                 imull
                                                          %eax, %eax
                                                                                 43
                                                                                           movl
                                                                                                   $0, %eax
      What are the memory addresses the local
                                        17
                                                 movl
                                                          %eax, -16(%ebp)
                                                                                 44 .L5:
      variables (a2, b, b2, c, c2) relative to the
                                                                                           leave
                                        18
                                                 movl
                                                          $1000, %eax
                                                                                 45
                                        19
                                                 subl
                                                          -4(%ebp), %eax
                                                                                 46
                                                                                           ret
      base pointer?
                                        20
                                                  subl
                                                          -8(%ebp), %eax
      a2 = EBP - 12
                                        21
                                                 movl
                                                          %eax, -20(%ebp)
                                        22
                                                 movl
                                                          -20(%ebp), %eax
      b = FBP - 8
                                        23
                                                 imull
                                                          %eax, %eax
      b2 = EBP - 16
                                        24
                                                 movl
                                                          %eax, -24(%ebp)
                                                          -12(%ebp), %edx
                                        25
                                                 movl
      c = EBP - 20
                                        26
                                                 movl
                                                          -16(%ebp), %eax
      c2 = EBP - 24
                                        27
                                                  addl
                                                          %edx, %eax
```

```
int main(void) {
                                                                                                %eax, -24(%ebp)
                                       1 main:
                                                                              28
                                                                                        cmpl
  for (int a = 1; _____) {
                                       2
                                                pushl
                                                        %ebp
                                                                              29
                                                                                        jne
                                                                                                 .L4
                                                                                                -4(%ebp), %eax
                                                movl
                                                        %esp, %ebp
                                                                              30
                                                                                        movl
    int a2 = ____;
   for (int b = _____) {
                                       4
                                                subl
                                                        $32, %esp
                                                                              31
                                                                                        imull
                                                                                                -8(%ebp), %eax
                                                        $1, -4(%ebp)
                                                                                                -20(%ebp), %eax
      int b2 = ____;
                                                movl
                                                                              32
                                                                                        imull
      int c = ____;
                                                                              33
                                                jmp
                                                        .L2
                                                                                        jmp
                                                                                                 .L5
      int c2 = ____;
                                       7 .L7:
                                                                              34 .L4:
      if (_____) {
                                                movl
                                                        -4(%ebp), %eax
                                                                              35
                                                                                        addl
                                                                                                $1, -8(%ebp)
                                       8
        return ____;
                                                imull
                                       9
                                                        %eax, %eax
                                                                              36 .L3:
                                       10
                                                movl
                                                        %eax, -12(%ebp)
                                                                              37
                                                                                                $666, -8(%ebp)
                                                                                        cmpl
                                                        -4(%ebp), %eax
                                       11
                                                movl
                                                                              38
                                                                                        jle
                                                                                                 .L6
                                       12
                                                movl
                                                        %eax, -8(%ebp)
                                                                              39
                                                                                        addl
                                                                                                $1, -4(%ebp)
                                       13
  return 0;
                                                        .L3
                                                                              40 .L2:
                                                jmp
                                                                                                $333, -4(%ebp)
                                       14 .L6:
                                                                              41
                                                                                        cmpl
                                       15
                                                movl
                                                        -8(%ebp), %eax
                                                                              42
                                                                                        jle
                                                                                                 .L7
                                       16
                                                imull
                                                        %eax, %eax
                                                                              43
                                                                                        movl
                                                                                                $0, %eax
      Fill in the missing code.
                                       17
                                                movl
                                                        %eax, -16(%ebp)
                                                                              44 .L5:
                                                                                        leave
                                       18
                                                movl
                                                        $1000, %eax
                                                                              45
                                       19
                                                subl
                                                        -4(%ebp), %eax
                                                                              46
                                                                                        ret
                                                        -8(%ebp), %eax
                                       20
                                                subl
                                       21
                                                movl
                                                        %eax, -20(%ebp)
                                       22
                                                movl
                                                        -20(%ebp), %eax
                                       23
                                                imull
                                                        %eax, %eax
                                       24
                                                movl
                                                        %eax, -24(%ebp)
                                                        -12(%ebp), %edx
                                       25
                                                movl
                                       26
                                                movl
                                                        -16(%ebp), %eax
                                       27
                                                addl
                                                        %edx, %eax
```

```
%eax, -24(%ebp)
int main(void) {
                                       1 main:
                                                                               28
                                                                                         cmpl
  for (int a = 1; a <= 333; a++) {
                                       2
                                                pushl
                                                        %ebp
                                                                               29
                                                                                         jne
                                                                                                  .L4
    int a2 = ____;
                                                                                                 -4(%ebp), %eax
                                       3
                                                movl
                                                        %esp, %ebp
                                                                               30
                                                                                         movl
    for (int b = ____) {
                                       4
                                                subl
                                                        $32, %esp
                                                                               31
                                                                                         imull
                                                                                                 -8(%ebp), %eax
                                                        $1, -4(%ebp)
                                                                                                 -20(%ebp), %eax
      int b2 = ____;
                                                movl
                                                                               32
                                                                                         imull
      int c = ____;
                                                                               33
                                                jmp
                                                         .L2
                                                                                         jmp
                                                                                                  .L5
     int c2 = ____;
                                       7 .L7:
                                                                               34 .L4:
      if (_____) {
                                                movl
                                                        -4(%ebp), %eax
                                                                               35
                                                                                         addl
                                                                                                 $1, -8(%ebp)
                                       8
        return ____;
                                                imull
                                       9
                                                        %eax, %eax
                                                                               36 .L3:
                                       10
                                                movl
                                                        %eax, -12(%ebp)
                                                                               37
                                                                                                 $666, -8(%ebp)
                                                                                         cmpl
                                                        -4(%ebp), %eax
                                       11
                                                movl
                                                                               38
                                                                                         jle
                                                                                                  .L6
                                       12
                                                movl
                                                        %eax, -8(%ebp)
                                                                               39
                                                                                         addl
                                                                                                 $1, -4(%ebp)
                                       13
  return 0;
                                                         .L3
                                                                               40 .L2:
                                                jmp
                                       14 .L6:
                                                                               41
                                                                                         cmpl
                                                                                                 $333, -4(%ebp)
                                       15
                                                movl
                                                        -8(%ebp), %eax
                                                                               42
                                                                                         jle
                                                                                                 .L7
                                       16
                                                imull
                                                        %eax, %eax
                                                                               43
                                                                                         movl
                                                                                                 $0, %eax
      Fill in the missing code.
                                       17
                                                movl
                                                        %eax, -16(%ebp)
                                                                               44 .L5:
                                                                                         leave
                                       18
                                                movl
                                                        $1000, %eax
                                                                               45
                                       19
                                                subl
                                                        -4(%ebp), %eax
                                                                               46
                                                                                         ret
                                       20
                                                subl
                                                        -8(%ebp), %eax
                                       21
                                                movl
                                                        %eax, -20(%ebp)
                                       22
                                                movl
                                                        -20(%ebp), %eax
                                       23
                                                imull
                                                        %eax, %eax
                                       24
                                                movl
                                                        %eax, -24(%ebp)
                                                        -12(%ebp), %edx
                                       25
                                                movl
                                       26
                                                movl
                                                        -16(%ebp), %eax
                                       27
                                                addl
                                                        %edx, %eax
```

```
int main(void) {
                                                                                                   %eax, -24(%ebp)
                                        1 main:
                                                                                28
                                                                                           cmpl
  for (int a = 1; a <= 333; a++) {
                                        2
                                                 pushl
                                                          %ebp
                                                                                29
                                                                                           jne
                                                                                                   .L4
                                                                                                   -4(%ebp), %eax
    int a2 = \mathbf{a} * \mathbf{a};
                                                 movl
                                                          %esp, %ebp
                                                                                30
                                                                                           movl
    for (int b = _____) {
                                        4
                                                 subl
                                                          $32, %esp
                                                                                31
                                                                                           imull
                                                                                                   -8(%ebp), %eax
                                                          $1, -4(%ebp)
                                                                                                   -20(%ebp), %eax
      int b2 = ____;
                                                 movl
                                                                                32
                                                                                           imull
      int c = ____;
                                                                                33
                                                 jmp
                                                          .L2
                                                                                           jmp
                                                                                                   .L5
      int c2 = ____;
                                        7 .L7:
                                                                                34 .L4:
      if (_____) {
                                                 movl
                                                          -4(%ebp), %eax
                                                                                35
                                                                                           addl
                                                                                                   $1, -8(%ebp)
                                        8
        return ____;
                                                 imull
                                                          %eax, %eax
                                        9
                                                                                36 .L3:
                                                         %eax, -12(%ebp)
                                        10
                                                 movl
                                                                                37
                                                                                                   $666, -8(%ebp)
                                                                                           cmpl
                                        11
                                                 movl
                                                          -4(%ebp), %eax
                                                                                38
                                                                                           jle
                                                                                                   .L6
                                        12
                                                 movl
                                                          %eax, -8(%ebp)
                                                                                39
                                                                                           addl
                                                                                                   $1, -4(%ebp)
                                        13
  return 0;
                                                          .L3
                                                                                40 .L2:
                                                 jmp
                                                                                                   $333, -4(%ebp)
                                        14 .L6:
                                                                                41
                                                                                           cmpl
                                        15
                                                 movl
                                                          -8(%ebp), %eax
                                                                                42
                                                                                           jle
                                                                                                   .L7
                                        16
                                                 imull
                                                         %eax, %eax
                                                                                43
                                                                                           movl
                                                                                                   $0, %eax
      Fill in the missing code.
                                        17
                                                 movl
                                                          %eax, -16(%ebp)
                                                                                44 .L5:
                                                                                           leave
                                        18
                                                 movl
                                                          $1000, %eax
                                                                                45
                                        19
                                                 subl
                                                          -4(%ebp), %eax
                                                                                46
                                                                                           ret
                                        20
                                                 subl
                                                          -8(%ebp), %eax
                                        21
                                                 movl
                                                          %eax, -20(%ebp)
                                        22
                                                 movl
                                                          -20(%ebp), %eax
                                        23
                                                 imull
                                                          %eax, %eax
                                        24
                                                 movl
                                                          %eax, -24(%ebp)
                                        25
                                                 movl
                                                          -12(%ebp), %edx
                                        26
                                                 movl
                                                          -16(%ebp), %eax
                                        27
                                                 addl
                                                          %edx, %eax
```

```
int main(void) {
                                                                                                   %eax, -24(%ebp)
                                        1 main:
                                                                                28
                                                                                          cmpl
  for (int a = 1; a <= 333; a++) {
                                        2
                                                 pushl
                                                         %ebp
                                                                                29
                                                                                          jne
                                                                                                   .L4
                                                                                                   -4(%ebp), %eax
    int a2 = a * a:
                                        3
                                                 movl
                                                         %esp, %ebp
                                                                                30
                                                                                          movl
    for (int b = a; b <= 666; b++) {
                                        4
                                                 subl
                                                         $32, %esp
                                                                                31
                                                                                          imull
                                                                                                   -8(%ebp), %eax
      int b2 = ____;
                                                         $1, -4(%ebp)
                                                 movl
                                                                                32
                                                                                          imull
                                                                                                   -20(%ebp), %eax
                                                                                33
      int c = ____;
                                                 jmp
                                                          .L2
                                                                                          jmp
                                                                                                   .L5
      int c2 = ____;
                                        7 .L7:
                                                                                34 .L4:
      if (_____) {
                                                 movl
                                                         -4(%ebp), %eax
                                                                                35
                                                                                          addl
                                                                                                   $1, -8(%ebp)
                                        8
        return ____;
                                                 imull
                                        9
                                                         %eax, %eax
                                                                                36 .L3:
                                        10
                                                 movl
                                                         %eax, -12(%ebp)
                                                                                37
                                                                                                   $666, -8(%ebp)
                                                                                          cmpl
                                        11
                                                 movl
                                                         -4(%ebp), %eax
                                                                                38
                                                                                          jle
                                                                                                   .L6
                                        12
                                                         %eax, -8(%ebp)
                                                                                39
                                                                                          addl
                                                                                                   $1, -4(%ebp)
                                                 movl
  return 0;
                                        13
                                                          .L3
                                                                                40 .L2:
                                                 jmp
                                        14 .L6:
                                                                                41
                                                                                          cmpl
                                                                                                   $333, -4(%ebp)
                                        15
                                                 movl
                                                         -8(%ebp), %eax
                                                                                42
                                                                                          jle
                                                                                                   .L7
                                        16
                                                 imull
                                                         %eax, %eax
                                                                                43
                                                                                          movl
                                                                                                   $0, %eax
      Fill in the missing code.
                                        17
                                                 movl
                                                         %eax, -16(%ebp)
                                                                                44 .L5:
                                        18
                                                 movl
                                                         $1000, %eax
                                                                                45
                                                                                          leave
                                        19
                                                 subl
                                                         -4(%ebp), %eax
                                                                                46
                                                                                          ret
                                        20
                                                 subl
                                                         -8(%ebp), %eax
                                        21
                                                 movl
                                                         %eax, -20(%ebp)
                                        22
                                                 movl
                                                         -20(%ebp), %eax
                                        23
                                                 imull
                                                         %eax, %eax
                                        24
                                                 movl
                                                         %eax, -24(%ebp)
                                        25
                                                 movl
                                                         -12(%ebp), %edx
                                        26
                                                 movl
                                                         -16(%ebp), %eax
                                        27
                                                 addl
                                                         %edx, %eax
```

```
int main(void) {
                                                                                                    %eax, -24(%ebp)
                                         1 main:
                                                                                 28
                                                                                            cmpl
  for (int a = 1; a <= 333; a++) {
                                         2
                                                  pushl
                                                          %ebp
                                                                                 29
                                                                                            jne
                                                                                                     .L4
                                                                                                    -4(%ebp), %eax
    int a2 = a * a:
                                         3
                                                  movl
                                                          %esp, %ebp
                                                                                 30
                                                                                            movl
    for (int b = a; b <= 666; b++) {
                                         4
                                                  subl
                                                          $32, %esp
                                                                                 31
                                                                                            imull
                                                                                                    -8(%ebp), %eax
      int b2 = \mathbf{b} * \mathbf{b};
                                                          $1, -4(%ebp)
                                                  movl
                                                                                 32
                                                                                            imull
                                                                                                    -20(%ebp), %eax
      int c = ____;
                                                  jmp
                                                           .L2
                                                                                 33
                                                                                            jmp
                                                                                                     .L5
      int c2 = ____;
                                         7 .L7:
                                                                                 34 .L4:
      if (_____) {
                                                  movl
                                                          -4(%ebp), %eax
                                                                                 35
                                                                                            addl
                                                                                                    $1, -8(%ebp)
                                         8
        return ____;
                                                  imull
                                         9
                                                          %eax, %eax
                                                                                 36 .L3:
                                         10
                                                  movl
                                                          %eax, -12(%ebp)
                                                                                 37
                                                                                                    $666, -8(%ebp)
                                                                                            cmpl
                                                          -4(%ebp), %eax
                                         11
                                                  movl
                                                                                 38
                                                                                            jle
                                                                                                     .L6
                                         12
                                                  movl
                                                          %eax, -8(%ebp)
                                                                                 39
                                                                                            addl
                                                                                                    $1, -4(%ebp)
                                         13
  return 0;
                                                           .L3
                                                                                 40 .L2:
                                                  jmp
                                         14 .L6:
                                                                                 41
                                                                                            cmpl
                                                                                                    $333, -4(%ebp)
                                        15
                                                  movl
                                                          -8(%ebp), %eax
                                                                                 42
                                                                                            jle
                                                                                                     .L7
                                         16
                                                  imull
                                                          %eax, %eax
                                                                                 43
                                                                                            movl
                                                                                                    $0, %eax
      Fill in the missing code.
                                         17
                                                  movl
                                                          %eax, -16(%ebp)
                                                                                 44 .L5:
                                                                                            leave
                                         18
                                                  movl
                                                          $1000, %eax
                                                                                 45
                                                          -4(%ebp), %eax
                                         19
                                                  subl
                                                                                 46
                                                                                            ret
                                         20
                                                  subl
                                                          -8(%ebp), %eax
                                         21
                                                  movl
                                                          %eax, -20(%ebp)
                                         22
                                                  movl
                                                          -20(%ebp), %eax
                                         23
                                                  imull
                                                          %eax, %eax
                                         24
                                                  movl
                                                          %eax, -24(%ebp)
                                         25
                                                  movl
                                                          -12(%ebp), %edx
                                         26
                                                  movl
                                                          -16(%ebp), %eax
                                         27
                                                  addl
                                                          %edx, %eax
```

```
int main(void) {
                                                                                                   %eax, -24(%ebp)
                                        1 main:
                                                                                 28
                                                                                           cmpl
  for (int a = 1; a <= 333; a++) {
                                        2
                                                 pushl
                                                          %ebp
                                                                                 29
                                                                                           jne
                                                                                                    .L4
                                                                                                   -4(%ebp), %eax
    int a2 = a * a:
                                        3
                                                 movl
                                                          %esp, %ebp
                                                                                 30
                                                                                           movl
    for (int b = a; b <= 666; b++) {
                                        4
                                                 subl
                                                          $32, %esp
                                                                                 31
                                                                                           imull
                                                                                                   -8(%ebp), %eax
                                                          $1, -4(%ebp)
      int b2 = b * b;
                                                 movl
                                                                                 32
                                                                                           imull
                                                                                                    -20(%ebp), %eax
      int c = 1000 - a - b;
                                                 jmp
                                                          .L2
                                                                                 33
                                                                                           jmp
                                                                                                    .L5
      int c2 = ____;
                                        7 .L7:
                                                                                 34 .L4:
      if (_____) {
                                                 movl
                                                          -4(%ebp), %eax
                                                                                 35
                                                                                           addl
                                                                                                   $1, -8(%ebp)
                                        8
                                                 imull
                                                          %eax, %eax
        return ____;
                                        9
                                                                                 36 .L3:
                                        10
                                                 movl
                                                          %eax, -12(%ebp)
                                                                                 37
                                                                                                   $666, -8(%ebp)
                                                                                           cmpl
                                        11
                                                 movl
                                                          -4(%ebp), %eax
                                                                                 38
                                                                                           jle
                                                                                                    .L6
                                        12
                                                 movl
                                                          %eax, -8(%ebp)
                                                                                 39
                                                                                           addl
                                                                                                   $1, -4(%ebp)
  return 0;
                                        13
                                                          .L3
                                                                                 40 .L2:
                                                 jmp
                                        14 .L6:
                                                                                 41
                                                                                           cmpl
                                                                                                   $333, -4(%ebp)
                                        15
                                                 movl
                                                          -8(%ebp), %eax
                                                                                 42
                                                                                           jle
                                                                                                    .L7
                                        16
                                                 imull
                                                          %eax, %eax
                                                                                 43
                                                                                           movl
                                                                                                   $0, %eax
      Fill in the missing code.
                                        17
                                                 movl
                                                          %eax, -16(%ebp)
                                                                                 44 .L5:
                                        18
                                                 movl
                                                          $1000, %eax
                                                                                 45
                                                                                           leave
                                        19
                                                 subl
                                                          -4(%ebp), %eax
                                                                                 46
                                                                                           ret
                                        20
                                                 subl
                                                          -8(%ebp), %eax
                                        21
                                                 movl
                                                          %eax, -20(%ebp)
                                        22
                                                 movl
                                                          -20(%ebp), %eax
                                        23
                                                 imull
                                                          %eax, %eax
                                        24
                                                 movl
                                                          %eax, -24(%ebp)
                                                          -12(%ebp), %edx
                                        25
                                                 movl
                                        26
                                                 movl
                                                          -16(%ebp), %eax
                                        27
                                                  addl
                                                          %edx, %eax
```

```
int main(void) {
                                                                                                   %eax, -24(%ebp)
                                        1 main:
                                                                                 28
                                                                                           cmpl
  for (int a = 1; a <= 333; a++) {
                                        2
                                                 pushl
                                                          %ebp
                                                                                 29
                                                                                           jne
                                                                                                    .L4
                                                                                                   -4(%ebp), %eax
    int a2 = a * a:
                                        3
                                                 movl
                                                          %esp, %ebp
                                                                                 30
                                                                                           movl
    for (int b = a; b <= 666; b++) {
                                        4
                                                 subl
                                                          $32, %esp
                                                                                 31
                                                                                           imull
                                                                                                   -8(%ebp), %eax
                                                          $1, -4(%ebp)
      int b2 = b * b;
                                                 movl
                                                                                 32
                                                                                           imull
                                                                                                   -20(%ebp), %eax
      int c = 1000 - a - b;
                                                 jmp
                                                          .L2
                                                                                 33
                                                                                           jmp
                                                                                                    .L5
      int c2 = c * c;
                                        7 .L7:
                                                                                 34 .L4:
      if (_____) {
                                                 movl
                                                          -4(%ebp), %eax
                                                                                 35
                                                                                           addl
                                                                                                   $1, -8(%ebp)
                                        8
                                                 imull
        return ____;
                                        9
                                                          %eax, %eax
                                                                                 36 .L3:
                                        10
                                                 movl
                                                          %eax, -12(%ebp)
                                                                                 37
                                                                                                   $666, -8(%ebp)
                                                                                           cmpl
                                        11
                                                 movl
                                                          -4(%ebp), %eax
                                                                                 38
                                                                                           jle
                                                                                                    .L6
                                        12
                                                 movl
                                                          %eax, -8(%ebp)
                                                                                 39
                                                                                           addl
                                                                                                   $1, -4(%ebp)
  return 0;
                                        13
                                                          .L3
                                                                                 40 .L2:
                                                 jmp
                                        14 .L6:
                                                                                 41
                                                                                           cmpl
                                                                                                   $333, -4(%ebp)
                                        15
                                                 movl
                                                          -8(%ebp), %eax
                                                                                 42
                                                                                           jle
                                                                                                    .L7
                                        16
                                                 imull
                                                          %eax, %eax
                                                                                 43
                                                                                           movl
                                                                                                   $0, %eax
      Fill in the missing code.
                                        17
                                                 movl
                                                          %eax, -16(%ebp)
                                                                                 44 .L5:
                                                                                           leave
                                        18
                                                 movl
                                                          $1000, %eax
                                                                                 45
                                                          -4(%ebp), %eax
                                        19
                                                 subl
                                                                                 46
                                                                                           ret
                                        20
                                                 subl
                                                          -8(%ebp), %eax
                                        21
                                                 movl
                                                          %eax, -20(%ebp)
                                        22
                                                 movl
                                                          -20(%ebp), %eax
                                        23
                                                 imull
                                                          %eax, %eax
                                        24
                                                 movl
                                                          %eax, -24(%ebp)
                                                          -12(%ebp), %edx
                                        25
                                                 movl
                                        26
                                                 movl
                                                          -16(%ebp), %eax
                                        27
                                                  addl
                                                          %edx, %eax
```

```
%eax, -24(%ebp)
int main(void) {
                                        1 main:
                                                                                 28
                                                                                            Cmpl
  for (int a = 1; a <= 333; a++) {
                                        2
                                                  pushl
                                                          %ebp
                                                                                 29
                                                                                            jne
                                                                                                    .L4
                                                                                                    -4(%ebp), %eax
    int a2 = a * a:
                                         3
                                                  movl
                                                          %esp, %ebp
                                                                                 30
                                                                                            movl
    for (int b = a; b <= 666; b++) {
                                         4
                                                  subl
                                                          $32, %esp
                                                                                 31
                                                                                            imull
                                                                                                    -8(%ebp), %eax
                                                          $1, -4(%ebp)
      int b2 = b * b;
                                                  movl
                                                                                 32
                                                                                            imull
                                                                                                    -20(%ebp), %eax
      int c = 1000 - a - b;
                                                  jmp
                                                           .L2
                                                                                 33
                                                                                            jmp
                                                                                                    .L5
      int c2 = c * c;
                                        7 .L7:
                                                                                 34 .L4:
      if (a2 + b2 == c2) {
                                                  movl
                                                          -4(%ebp), %eax
                                                                                 35
                                                                                            addl
                                                                                                    $1, -8(%ebp)
                                                  imull
                                                          %eax, %eax
        return _____;
                                         9
                                                                                 36 .L3:
                                        10
                                                  movl
                                                          %eax, -12(%ebp)
                                                                                 37
                                                                                                    $666, -8(%ebp)
                                                                                            cmpl
                                        11
                                                  movl
                                                          -4(%ebp), %eax
                                                                                 38
                                                                                            jle
                                                                                                    .L6
                                        12
                                                  movl
                                                          %eax, -8(%ebp)
                                                                                 39
                                                                                            addl
                                                                                                    $1, -4(%ebp)
  return 0;
                                        13
                                                           .L3
                                                                                 40 .L2:
                                                  jmp
                                        14 .L6:
                                                                                 41
                                                                                            cmpl
                                                                                                    $333, -4(%ebp)
                                        15
                                                  movl
                                                          -8(%ebp), %eax
                                                                                 42
                                                                                            jle
                                                                                                    .L7
                                        16
                                                  imull
                                                          %eax, %eax
                                                                                 43
                                                                                            movl
                                                                                                    $0, %eax
      Fill in the missing code.
                                        17
                                                  movl
                                                          %eax, -16(%ebp)
                                                                                 44 .L5:
                                        18
                                                  movl
                                                          $1000, %eax
                                                                                 45
                                                                                            leave
                                        19
                                                  subl
                                                          -4(%ebp), %eax
                                                                                 46
                                                                                            ret
                                        20
                                                  subl
                                                          -8(%ebp), %eax
                                        21
                                                  movl
                                                          %eax, -20(%ebp)
                                        22
                                                  movl
                                                          -20(%ebp), %eax
                                        23
                                                  imull
                                                          %eax, %eax
                                        24
                                                  movl
                                                          %eax, -24(%ebp)
                                        25
                                                  movl
                                                          -12(%ebp), %edx
                                        26
                                                  movl
                                                          -16(%ebp), %eax
                                        27
                                                  addl
                                                          %edx, %eax
```

```
%eax, -24(%ebp)
int main(void) {
                                         1 main:
                                                                                  28
                                                                                             cmpl
  for (int a = 1; a <= 333; a++) {
                                         2
                                                  pushl
                                                           %ebp
                                                                                  29
                                                                                             jne
                                                                                                      .L4
                                                                                                      -4(%ebp), %eax
    int a2 = a * a:
                                         3
                                                  movl
                                                           %esp, %ebp
                                                                                  30
                                                                                             movl
    for (int b = a; b <= 666; b++) {
                                         4
                                                   subl
                                                           $32, %esp
                                                                                  31
                                                                                             imull
                                                                                                      -8(%ebp), %eax
      int b2 = b * b;
                                         5
                                                  movl
                                                           $1, -4(%ebp)
                                                                                  32
                                                                                             imull
                                                                                                      -20(%ebp), %eax
      int c = 1000 - a - b;
                                                                                  33
                                                  jmp
                                                           .L2
                                                                                             jmp
                                                                                                      .L5
      int c2 = c * c;
                                         7 .L7:
                                                                                  34 .L4:
      if (a2 + b2 == c2) {
                                                  movl
                                                           -4(%ebp), %eax
                                                                                  35
                                                                                             addl
                                                                                                     $1, -8(%ebp)
                                         8
        return a * b * c;
                                                  imull
                                                           %eax, %eax
                                                                                  36 .L3:
                                         9
                                                  movl
                                                           %eax, -12(%ebp)
                                                                                  37
                                                                                                      $666, -8(%ebp)
                                         10
                                                                                             cmpl
                                         11
                                                  movl
                                                           -4(%ebp), %eax
                                                                                  38
                                                                                             jle
                                                                                                      .L6
                                         12
                                                  movl
                                                           %eax, -8(%ebp)
                                                                                  39
                                                                                             addl
                                                                                                     $1, -4(%ebp)
                                         13
  return 0;
                                                           .L3
                                                                                  40 .L2:
                                                  jmp
                                         14 .L6:
                                                                                  41
                                                                                             cmpl
                                                                                                     $333, -4(%ebp)
                                         15
                                                  movl
                                                           -8(%ebp), %eax
                                                                                  42
                                                                                             jle
                                                                                                      .L7
                                         16
                                                  imull
                                                           %eax, %eax
                                                                                  43
                                                                                             movl
                                                                                                     $0, %eax
      Fill in the missing code.
                                         17
                                                  movl
                                                           %eax, -16(%ebp)
                                                                                  44 .L5:
                                                                                             leave
                                         18
                                                  movl
                                                           $1000, %eax
                                                                                  45
                                         19
                                                  subl
                                                           -4(%ebp), %eax
                                                                                  46
                                                                                             ret
                                         20
                                                  subl
                                                           -8(%ebp), %eax
                                         21
                                                  movl
                                                           %eax, -20(%ebp)
                                         22
                                                  movl
                                                           -20(%ebp), %eax
                                         23
                                                  imull
                                                           %eax, %eax
                                         24
                                                  movl
                                                           %eax, -24(%ebp)
                                                           -12(%ebp), %edx
                                         25
                                                  movl
                                         26
                                                  movl
                                                           -16(%ebp), %eax
                                         27
                                                   addl
                                                           %edx, %eax
```

Stack Frame

```
call.s
                                         27
                                                    pushl
                                                             $3
                                         28
                                                    call
                                                             bar
1 p:
                                                    addl
                                                             $16, %esp
2
          .zero
                   4
                                         29
                                                    addl
3 bar:
                                         30
                                                             %ebx, %eax
          pushl
                  %ebp
                                         31
                                                    movl
                                                             %eax, p
4
          movl
                  %esp, %ebp
5
                                         32
                                                    nop
6
          subl
                  $16, %esp
                                         33
                                                    movl
                                                             -4(%ebp), %ebx
                  8(%ebp), %edx
          movl
                                         34
                                                    leave
                  12(%ebp), %eax
          movl
                                         35
                                                    ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                   Which lines of code correspond to a
10
          movl
                  %eax, -4(%ebp)
11
                                                   caller/callee prologue?
                  -4(%ebp), %eax
          movl
12
          addl
13
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
                  (%edx,%eax), %ebx
23
          leal
          subl
                  $4, %esp
24
25
          pushl
                  $5
          pushl
                   $4
26
```

Stack Frame

```
call.s
                                                        pushl
                                            27
                                                                 $3
                                            28
                                                        call
                                                                 bar
1 p:
                                                                 $16, %esp
                                                        addl
2
           .zero
                    4
                                            29
                                                        addl
3 bar:
                                                                 %ebx, %eax
                                            30
                    %ebp
           pushl
                                            31
                                                        movl
                                                                 %eax, p
4
                   %esp, %ebp
5
          movl
                                            32
                                                        nop
           subl
                    $16, %esp
                                            33
                                                        movl
                                                                 -4(%ebp), %ebx
6
                    8(%ebp), %edx
          movl
                                            34
                                                        leave
                   12(%ebp), %eax
                                                        ret
          movl
                                            35
8
           addl
                   %edx, %eax
9
           subl
                   16(%ebp), %eax
                                                       Which lines of code correspond to a
10
          movl
                   %eax, -4(%ebp)
11
                                                       caller/callee prologue?
                   -4(%ebp), %eax
          movl
12
                                                       foo/bar as caller/callee
           addl
                    $1, %eax
13
          leave
14
                                                        24-27 = caller prologue
15
          ret
16 foo:
                                                        29 = caller epilogue
                    %ebp
           pushl
17
                                                        4-6 = callee prologue
                    %esp, %ebp
          movl
18
          pushl
                    %ebx
                                                        13-15 = callee epilogue
19
20
           subl
                    $4, %esp
                                                       foo as callee
           movl
                    8(%ebp), %edx
21
                                                        17-20 = callee prologue
          movl
                   12(%ebp), %eax
22
                   (%edx,%eax), %ebx
23
           leal
                                                        33-35 = callee epilogue
           subl
                    $4, %esp
24
25
           pushl
                    $5
          pushl
                    $4
26
```

```
call.s
                                         27
                                                    pushl
                                                            $3
                                         28
                                                    call
                                                            bar
1 p:
                                                    addl
                                                            $16, %esp
2
          .zero
                  4
                                         29
                                                    addl
3 bar:
                                         30
                                                            %ebx, %eax
          pushl
                  %ebp
                                         31
                                                    movl
                                                            %eax, p
4
          movl
                  %esp, %ebp
5
                                         32
                                                    nop
6
          subl
                  $16, %esp
                                         33
                                                    movl
                                                            -4(%ebp), %ebx
                  8(%ebp), %edx
7
          movl
                                         34
                                                    leave
                  12(%ebp), %eax
          movl
                                         35
                                                    ret
8
          addl
                  %edx, %eax
9
          subl
                 16(%ebp), %eax
                                                   What does line 19 do in call.s? Why is
10
          movl
                 %eax, -4(%ebp)
11
                                                   it necessary?
                  -4(%ebp), %eax
          movl
12
13
          addl
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                  $4, %esp
          movl
                  8(%ebp), %edx
21
                  12(%ebp), %eax
22
          movl
                  (%edx,%eax), %ebx
23
          leal
          subl
                  $4, %esp
24
25
          pushl
                  $5
          pushl
                   $4
26
```

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                   %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                   $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                   8(%ebp), %edx
          movl
                                          34
                                                     leave
                  12(%ebp), %eax
          movl
                                          35
                                                      ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    What does line 19 do in call.s? Why is
10
          movl
                 %eax, -4(%ebp)
11
                                                     it necessary?
                  -4(%ebp), %eax
          movl
12
                                                     Saves EBX register since it's callee-saved
13
          addl
                   $1, %eax
          leave
14
                                                     and foo uses it. Doesn't matter that bar
15
          ret
16 foo:
                                                     never uses EBX.
          pushl
                   %ebp
17
          movl
                   %esp, %ebp
18
          pushl
                   %ebx
19
20
          subl
                   $4, %esp
          movl
                   8(%ebp), %edx
21
22
          movl
                   12(%ebp), %eax
                   (%edx,%eax), %ebx
23
          leal
                   $4, %esp
          subl
24
25
          pushl
                   $5
          pushl
                   $4
26
```

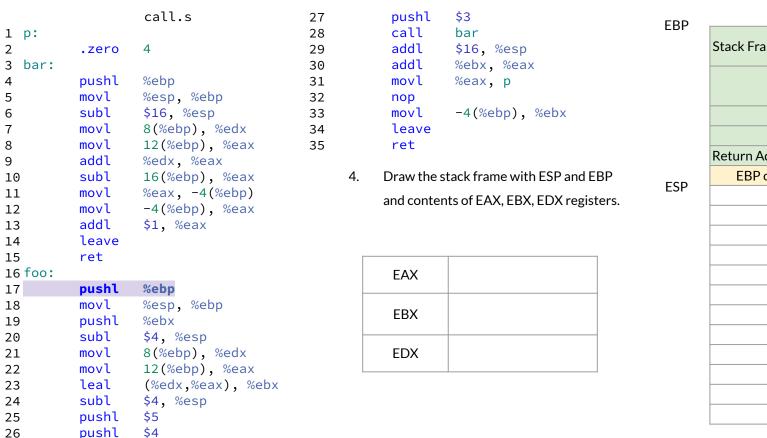
```
call.s
                                         27
                                                    pushl
                                                             $3
                                         28
                                                    call
                                                             bar
1 p:
                                                    addl
                                                             $16, %esp
2
          .zero
                   4
                                         29
                                                    addl
3 bar:
                                         30
                                                             %ebx, %eax
          pushl
                  %ebp
                                         31
                                                    movl
                                                             %eax, p
4
          movl
                  %esp, %ebp
5
                                         32
                                                    nop
6
          subl
                  $16, %esp
                                         33
                                                    movl
                                                             -4(%ebp), %ebx
                  8(%ebp), %edx
          movl
                                         34
                                                    leave
                  12(%ebp), %eax
          movl
                                         35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                   Why is EDX not saved by foo before
10
          movl
                  %eax, -4(%ebp)
11
                                                   calling bar despite the register being
                  -4(%ebp), %eax
          movl
12
                                                   overwritten in bar?
13
          addl
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
          movl
                  8(%ebp), %edx
22
          movl
                  12(%ebp), %eax
                  (%edx,%eax), %ebx
23
          leal
          subl
                  $4, %esp
24
25
          pushl
                  $5
          pushl
                   $4
26
```

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                  8(%ebp), %edx
          movl
                                          34
                                                     leave
                  12(%ebp), %eax
          movl
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Why is EDX not saved by foo before
10
                 %eax, -4(%ebp)
          movl
11
                                                    calling bar despite the register being
                  -4(%ebp), %eax
          movl
12
                                                    overwritten in bar?
13
          addl
                   $1, %eax
          leave
14
                                                    EDX does not need to persist after the
15
          ret
16 foo:
                                                    function call.
          pushl
                   %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                   $4, %esp
21
          movl
                  8(%ebp), %edx
22
          movl
                  12(%ebp), %eax
          leal
                   (%edx,%eax), %ebx
23
                   $4, %esp
          subl
24
25
          pushl
                   $5
          pushl
                   $4
26
```

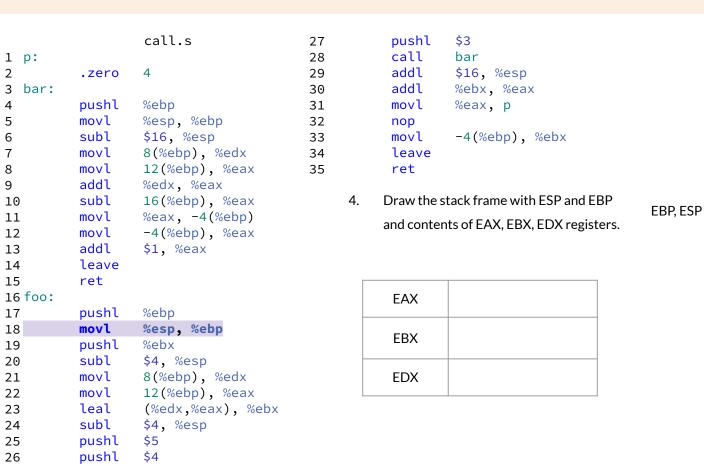
| | | call.s | 27 | | pushl | \$3 | EDD | |
|---------|-------|-------------------|----|----|---------------|------------------------------|------|------------|
| 1 p: | | | 28 | | call | bar | EBP | |
| 2 | .zero | 4 | 29 | | addl | \$16, %esp | | Stack Fram |
| 3 bar: | | | 30 | | addl | %ebx, %eax | ESP | |
| 4 | pushl | %ebp | 31 | | movl | %eax, p | | |
| 5 | movl | %esp, %ebp | 32 | | nop | | | |
| 6 | subl | \$16, %esp | 33 | | movl | -4(%ebp), %ebx | | |
| 7 | movl | 8(%ebp), %edx | 34 | | leave | | | |
| 8 | movl | 12(%ebp), %eax | 35 | | ret | | | |
| 9 | addl | %edx, %eax | | | | | | |
| 10 | subl | 16(%ebp), %eax | | 4. | Draw the | tack frame with ESP and EE | 3P | |
| 11 | movl | %eax, -4(%ebp) | | | and conta | ata af EAV EDV EDV vaciate | | |
| 12 | movl | -4(%ebp), %eax | | | and conte | nts of EAX, EBX, EDX registe | ers. | |
| 13 | addl | \$1, %eax | | | | | | |
| 14 | leave | | | | | | | |
| 15 | ret | | | | | | | |
| 16 foo: | | | | | EAX | | | |
| 17 | pushl | %ebp | | | | | | |
| 18 | movl | %esp, %ebp | | | ED) (| | | |
| 19 | pushl | %ebx | | | EBX | | | |
| 20 | subl | \$4, %esp | | | | | | |
| 21 | movl | 8(%ebp), %edx | | | EDX | | | |
| 22 | movl | 12(%ebp), %eax | | | | | | |
| 23 | leal | (%edx,%eax), %ebx | | | | | | |
| 24 | subl | \$4, %esp | | | | | | |
| 25 | pushl | \$5 | | | | | | |
| 26 | pushl | \$4 | | | | | | |

| Stack Frame of foo's Caller |
|-----------------------------|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

| | | call.s | 27 | pushl | \$3 | ED C | |
|---------|-------|-----------------------|----|---------------|--------------------------------|-------------|-----------------------------|
| 1 p: | | | 28 | call | bar | EBP | |
| 2 | .zero | 4 | 29 | addl | \$16, %esp | | Stack Frame of foo's Caller |
| 3 bar: | | | 30 | addl | %ebx, %eax | | |
| 4 | pushl | %ebp | 31 | movl | %eax, p | | Padding |
| 5 | movl | %esp, %ebp | 32 | nop | | | i adding |
| 6 | subl | \$16, %esp | 33 | movl | -4(%ebp), %ebx | | b |
| 7 | movl | 8(%ebp), %edx | 34 | leave | | | a |
| 8 | movl | 12(%ebp), %eax | 35 | ret | | | |
| 9 | addl | <pre>%edx, %eax</pre> | | | | ESP | Return Addr of foo's Caller |
| 10 | subl | 16(%ebp), %eax | 2 | 4. Draw the s | tack frame with ESP and EBP | | |
| 11 | movl | %eax, -4(%ebp) | | and conter | its of EAX, EBX, EDX registers | i. | |
| 12 | movl | -4(%ebp), %eax | | GG. 5555. | | • | |
| 13 | addl | \$1, %eax | | | | | |
| 14 | leave | | | | | | |
| 15 | ret | | | | | | |
| 16 foo: | | 0.1 | | EAX | | | |
| 17 | pushl | %ebp | | | | | |
| 18 | movl | %esp, %ebp | | EBX | | | |
| 19 | pushl | %ebx | | | | | |
| 20 | subl | \$4, %esp | | EDV | | | |
| 21 | movl | 8(%ebp), %edx | | EDX | | | |
| 22 | movl | 12(%ebp), %eax | | | | | |
| 23 | leal | (%edx,%eax), %ebx | | | | | |
| 24 | subl | \$4, %esp | | | | | |
| 25 | pushl | \$5 | | | | | |
| 26 | pushl | \$4 | | | | | |



Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller



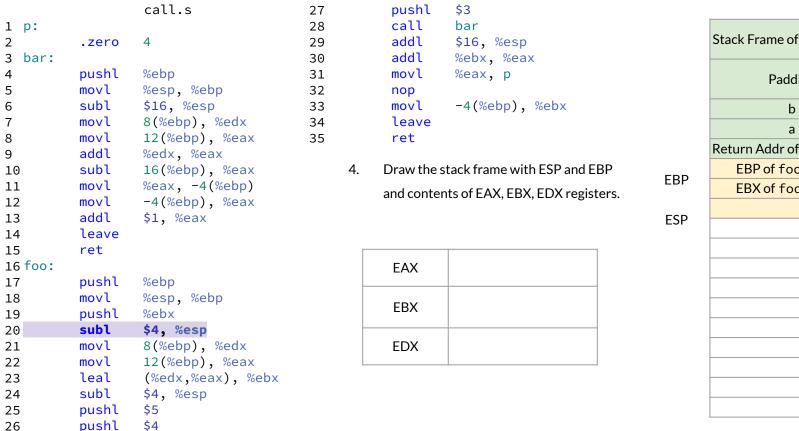
| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

26

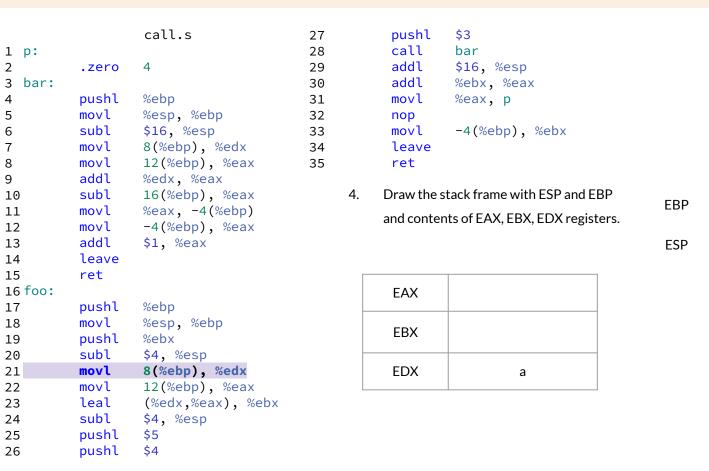
```
call.s
                                         27
                                                     pushl
                                                             $3
                                         28
                                                     call
                                                             bar
1 p:
                                                     addl
                                                             $16, %esp
2
          .zero
                   4
                                         29
                                                     addl
3 bar:
                                          30
                                                             %ebx, %eax
          pushl
                   %ebp
                                         31
                                                     movl
                                                             %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                         33
                                                     movl
                                                             -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                         34
          movl
                  12(%ebp), %eax
                                         35
                                                     ret
8
          addl
                  %edx, %eax
9
                                                    Draw the stack frame with FSP and FBP
          subl
                  16(%ebp), %eax
10
                  %eax, -4(%ebp)
          movl
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
          addl
13
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
          pushl
17
                   %ebp
          movl
                  %esp, %ebp
18
                                                     EBX
          pushl
                   %ebx
19
          subl
20
                   $4, %esp
          movl
                  8(%ebp), %edx
                                                     EDX
21
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
24
          subl
                   $4, %esp
25
          pushl
                   $5
          pushl
                   $4
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

EBP



Stack Frame of foo's Caller **Padding** Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller



| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
                                          29
          .zero
                   4
3 bar:
                                                     addl
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                   8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Draw the stack frame with FSP and FBP
10
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
          addl
13
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                       b
          pushl
17
                   %ebp
          movl
                  %esp, %ebp
18
                                                     EBX
          pushl
                   %ebx
19
          subl
20
                   $4, %esp
21
          movl
                   8(%ebp), %edx
                                                     EDX
                                                                       а
                   12(%ebp), %eax
22
          movl
          leal
                   (%edx,%eax), %ebx
23
24
          subl
                   $4, %esp
25
          pushl
                   $5
          pushl
                   $4
26
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

EBP

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                   %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                   $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                   8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                   12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Draw the stack frame with FSP and FBP
10
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                   -4(%ebp), %eax
          movl
12
          addl
13
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                       b
          pushl
17
                   %ebp
          movl
                   %esp, %ebp
18
                                                     EBX
                                                                     a + b
          pushl
                   %ebx
19
          subl
20
                   $4, %esp
21
          movl
                   8(%ebp), %edx
                                                     EDX
                                                                       а
                   12(%ebp), %eax
22
          movl
          leal
                   (%edx,%eax), %ebx
23
24
          subl
                   $4, %esp
25
          pushl
                   $5
          pushl
                   $4
26
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

EBP

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Draw the stack frame with FSP and FBP
10
                  %eax, -4(%ebp)
          movl
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
          addl
13
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                       b
          pushl
17
                   %ebp
                  %esp, %ebp
          movl
18
                                                     EBX
                                                                     a + b
          pushl
                   %ebx
19
          subl
20
                   $4, %esp
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                       а
                  12(%ebp), %eax
22
          movl
                  (%edx,%eax), %ebx
          leal
23
                   $4, %esp
24
          subl
25
          pushl
                   $5
          pushl
                   $4
26
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

EBP

```
call.s
1 p:
2
          .zero
                  4
3 bar:
          pushl
                  %ebp
4
          movl
                  %esp, %ebp
5
6
          subl
                  $16, %esp
                  8(%ebp), %edx
          movl
          movl
                  12(%ebp), %eax
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
                  -4(%ebp), %eax
          movl
12
          addl
13
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
                  $4, %esp
20
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                 (%edx,%eax), %ebx
23
                  $4, %esp
24
          subl
                  $5
25
          pushl
          pushl
                  $4
26
```

| pushl call addl addl movl nop | \$3 bar \$16, %esp %ebx, %eax %eax, p |
|--|---|
| movl leave ret | -4(%ebp), %ebx |

27 28

29

30

31

32

33

34

35

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | a |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| |
| |
| |
| |
| |
| |
| |

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
                                        30
          pushl
                  %ebp
                                        31
4
          movl
                  %esp, %ebp
5
                                        32
6
          subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
          movl
                                        34
          movl
                  12(%ebp), %eax
                                        35
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
          movl
                  -4(%ebp), %eax
12
          addl
                  $1, %eax
13
          leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
                  $4, %esp
20
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                  $4, %esp
24
          subl
25
          pushl
                  $5
26
          pushl
```

| pushl | \$3 |
|-----------------------------|----------------|
| call | bar |
| addl | \$16, %esp |
| addl | %ebx, %eax |
| movl | %eax, p |
| nop movl leave ret | -4(%ebp), %ebx |

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | a |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| |
| |
| |
| |
| |

EBP

| | | call.s | 27 | | pushl | |
|---------|-------|-----------------------|----|----|-------------|---|
| 1 p: | | | 28 | | call | |
| 2 | .zero | 4 | 29 | | addl | |
| 3 bar: | | | 30 | | addl | |
| 4 | pushl | %ebp | 31 | | movl | |
| 5 | movl | %esp, %ebp | 32 | | nop | |
| 6 | subl | \$16, %esp | 33 | | movl | |
| 7 | movl | 8(%ebp), %edx | 34 | | leave | |
| 8 | movl | 12(%ebp), %eax | 35 | | ret | |
| 9 | addl | <pre>%edx, %eax</pre> | | | | |
| 10 | subl | 16(%ebp), %eax | | 4. | Draw the st | į |
| 11 | movl | %eax, -4(%ebp) | | | and conten | + |
| 12 | movl | -4(%ebp), %eax | | | and conten | ι |
| 13 | addl | \$1, %eax | | | | |
| 14 | leave | | | | | |
| 15 | ret | | | | | Γ |
| 16 foo: | | | | | EAX | |
| 17 | pushl | %ebp | | | | L |
| 18 | movl | %esp, %ebp | | | EDV | |
| 19 | pushl | %ebx | | | EBX | |
| 20 | subl | \$4, %esp | | - | | r |
| 21 | movl | 8(%ebp), %edx | | | EDX | |
| 22 | movl | 12(%ebp), %eax | | L | | L |
| 23 | leal | (%edx,%eax), %ebx | | | | |
| 24 | subl | \$4, %esp | | | | |
| 25 | pushl | \$5 | | | | |
| 26 | pushl | \$4 | | | | |

| 27 | pushl | \$3 |
|----|-------|-----------------------|
| 28 | call | bar |
| 29 | addl | \$16, %esp |
| 30 | addl | <pre>%ebx, %eax</pre> |
| 31 | movl | %eax, p |
| 32 | nop | |
| 33 | movl | -4(%ebp), %ebx |
| 34 | leave | |
| 35 | ret | |

tack frame with ESP and EBP nts of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | а |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| |
| |
| |
| |
| |

EBP

| 1 p: | | call.s | 27 28 | | pushl call | \$3 bar |
|---------|-------|-------------------|----------|----|-------------|--------------------------------|
| 2 | .zero | 4 | 29 | | addl | \$16, %esp |
| 3 bar: | | | 30 | | addl | %ebx, %eax |
| 4 | pushl | %ebp | 31 | | movl | %eax, p |
| 5 | movl | %esp, %ebp | 32 | | nop_ | |
| 6 | subl | \$16, %esp | 33 | | movl | -4(%ebp), %ebx |
| 7 | movl | | 34 | | leave | |
| 8 | movl | 12(%ebp), %eax | 35 | | ret | |
| 9 | addl | %edx, %eax | | | | |
| 10 | subl | 16(%ebp), %eax | | 4. | Draw the st | tack frame with ESP and EBP |
| 11 | | %eax, -4(%ebp) | | | and conten | ts of EAX, EBX, EDX registers. |
| 12 | | -4(%ebp), %eax | | | | , , , |
| 13 | | \$1, %eax | | | | |
| 14 | leave | | | | | |
| 15 | ret | | | | | |
| 16 foo: | | | | | EAX | b |
| 17 | pushl | %ebp | | | | |
| 18 | movl | %esp, %ebp | | | EBX | a+b |
| 19 | pushl | %ebx | | | LDA | |
| 20 | subl | \$4, %esp | | | | |
| 21 | | 8(%ebp), %edx | | | EDX | a |
| 22 | movl | 12(%ebp), %eax | | | | |
| 23 | | (%edx,%eax), %ebx | | | | |
| 24 | subl | \$4, %esp | | | | |
| 25 | pushl | \$5 | | | | |
| 26 | pushl | \$4 | | | | |

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller 5 4 3 Return Addr of foo

EBP

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                             $16, %esp
                                                     addl
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                             %ebx, %eax
                   %ebp
          pushl
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                   %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                   $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                   8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
                                                    Draw the stack frame with FSP and FBP
          subl
                  16(%ebp), %eax
10
                                                                                           EBP
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
13
          addl
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                       b
          pushl
                   %ebp
17
          movl
                   %esp, %ebp
18
                                                     FBX
                                                                     a + b
          pushl
                   %ebx
19
          subl
                   $4, %esp
20
                                                                                           ESP
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                       а
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
          subl
                   $4, %esp
24
25
          pushl
                   $5
          pushl
                   $4
26
```

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller 5 4 3 Return Addr of foo EBP of foo

```
call.s
1 p:
2
          .zero
                  4
3 bar:
         pushl
                  %ebp
4
         movl
                  %esp, %ebp
5
6
          subl
                  $16, %esp
                  8(%ebp), %edx
         movl
                  12(%ebp), %eax
         movl
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
         movl
                %eax, -4(%ebp)
11
                 -4(%ebp), %eax
         movl
12
13
          addl
                  $1, %eax
         leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
         movl
                  %esp, %ebp
18
         pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
         movl
                  8(%ebp), %edx
                 12(%ebp), %eax
22
         movl
         leal
                 (%edx,%eax), %ebx
23
                  $4, %esp
          subl
24
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl $3
call bar
addl $16, %esp
addl %ebx, %eax
movl %eax, p
nop
movl -4(%ebp), %ebx
leave
ret
```

27

28

29

30

31

32

33

34

35

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | a |

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller 5 4 3 Return Addr of foo EBP of foo

EBP, ESP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
                                        30
         pushl
                  %ebp
                                        31
4
         movl
                  %esp, %ebp
5
                                        32
6
         subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
         movl
                                        34
                  12(%ebp), %eax
         movl
                                        35
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
         movl
                 %eax, -4(%ebp)
11
                 -4(%ebp), %eax
         movl
12
13
          addl
                  $1, %eax
         leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
         movl
                  %esp, %ebp
18
         pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
         movl
                 8(%ebp), %edx
                 12(%ebp), %eax
22
         movl
                 (%edx,%eax), %ebx
23
         leal
                  $4, %esp
24
          subl
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl $3
call bar
addl $16, %esp
addl %ebx, %eax
movl %eax, p
nop
movl -4(%ebp), %ebx
leave
ret
```

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | a |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| |
| |
| |
| |

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
                                        30
          pushl
                  %ebp
                                        31
4
          movl
                  %esp, %ebp
5
                                        32
6
          subl
                  $16, %esp
                                        33
7
                  8(%ebp), %edx
          movl
                                        34
                  12(%ebp), %eax
          movl
                                        35
8
          addl
                  %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
                  -4(%ebp), %eax
          movl
12
13
          addl
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
24
          subl
                  $4, %esp
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl $3
call bar
addl $16, %esp
addl %ebx, %eax
movl %eax, p
nop
movl -4(%ebp), %ebx
leave
ret
```

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | b |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| |
| |
| |
| |

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
                                        30
          pushl
                  %ebp
                                        31
4
          movl
                  %esp, %ebp
5
                                        32
6
          subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
          movl
                                        34
                  12(%ebp), %eax
8
          movl
                                        35
          addl
                  %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
                  -4(%ebp), %eax
          movl
12
13
          addl
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                  $4, %esp
24
          subl
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl $3
call bar
addl $16, %esp
addl %ebx, %eax
movl %eax, p
nop
movl -4(%ebp), %ebx
leave
ret
```

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 4 |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| |
| |
| |
| |

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
         pushl
                  %ebp
                                        31
4
         movl
                  %esp, %ebp
5
6
          subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
         movl
                                        34
                  12(%ebp), %eax
         movl
                                        35
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
10
         movl
                 %eax, -4(%ebp)
11
                  -4(%ebp), %eax
         movl
12
13
          addl
                  $1, %eax
         leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
                  %esp, %ebp
         movl
18
         pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
         movl
                 8(%ebp), %edx
                 12(%ebp), %eax
22
         movl
                 (%edx,%eax), %ebx
23
         leal
          subl
                  $4, %esp
24
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl
        $3
call
        bar
addl
        $16, %esp
addl
        %ebx, %eax
movl
        %eax, p
nop
movl
        -4(%ebp), %ebx
leave
ret
```

30

32

Draw the stack frame with FSP and FBP and contents of EAX, EBX, EDX registers.

| EAX | 7 |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| |
| |
| |
| |

EBP

```
call.s
1 p:
2
          .zero
3 bar:
          pushl
                  %ebp
4
          movl
                  %esp, %ebp
5
6
          subl
                  $16, %esp
                  8(%ebp), %edx
          movl
          movl
                 12(%ebp), %eax
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
          movl
                 -4(%ebp), %eax
12
          addl
                  $1, %eax
13
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
                  $4, %esp
20
21
          movl
                 8(%ebp), %edx
                 12(%ebp), %eax
22
          movl
          leal
                 (%edx,%eax), %ebx
23
                  $4, %esp
24
          subl
25
          pushl
                  $5
          pushl
                  $4
26
```

| pushl | \$3 |
|-------|----------------|
| call | bar |
| addl | \$16, %esp |
| addl | %ebx, %eax |
| movl | %eax, p |
| nop | |
| movl | -4(%ebp), %ebx |
| leave | |
| ret | |

27 28

29

30

31

32

33

34

35

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 2 |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| |
| |
| |
| |

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                                        29
3 bar:
                                        30
          pushl
                  %ebp
4
                                        31
          movl
                  %esp, %ebp
5
                                        32
6
          subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
          movl
                                        34
          movl
                  12(%ebp), %eax
                                        35
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                  %eax, -4(%ebp)
11
          movl
                  -4(%ebp), %eax
12
          addl
13
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
                  $4, %esp
20
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                 (%edx,%eax), %ebx
23
                  $4, %esp
24
          subl
25
          pushl
                  $5
          pushl
                  $4
26
```

| pushl | \$3 |
|-------|-----------------------|
| call | bar |
| addl | \$16, %esp |
| addl | <pre>%ebx, %eax</pre> |
| movl | %eax, p |
| nop | |
| movl | -4(%ebp), %ebx |
| leave | |
| ret | |

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 2 |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| 2 |
| |
| |
| |

ESP

EBP

```
call.s
                                        27
                                        28
1 p:
2
          .zero
                  4
                                        29
3 bar:
                                        30
         pushl
                  %ebp
                                        31
4
         movl
                  %esp, %ebp
5
                                        32
6
          subl
                  $16, %esp
                                        33
                  8(%ebp), %edx
         movl
                                        34
                  12(%ebp), %eax
         movl
                                        35
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
         movl
                 %eax, -4(%ebp)
11
                  -4(%ebp), %eax
12
          movl
13
          addl
                  $1, %eax
          leave
14
15
          ret
16 foo:
          pushl
17
                  %ebp
                  %esp, %ebp
         movl
18
         pushl
                  %ebx
19
          subl
20
                  $4, %esp
21
         movl
                  8(%ebp), %edx
                 12(%ebp), %eax
22
         movl
                 (%edx,%eax), %ebx
23
         leal
                  $4, %esp
24
          subl
25
          pushl
                  $5
          pushl
                  $4
26
```

```
pushl $3
call bar
addl $16, %esp
addl %ebx, %eax
movl %eax, p
nop
movl -4(%ebp), %ebx
leave
ret
```

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 2 |
|-----|-------|
| EBX | a + b |
| EDX | 3 |

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| 2 |
| |
| |
| |

EBP

```
call.s
                                         27
                                                     pushl
                                                             $3
                                          28
                                                     call
                                                             bar
1 p:
                                                     addl
                                                             $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                             %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                             %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                             -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Draw the stack frame with FSP and FBP
10
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                   -4(%ebp), %eax
          movl
12
13
          addl
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                                      3
                                                     EAX
          pushl
                   %ebp
17
          movl
                  %esp, %ebp
18
                                                     EBX
                                                                     a + b
          pushl
                   %ebx
19
          subl
                   $4, %esp
20
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                      3
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                   $4, %esp
24
          subl
25
          pushl
                   $5
          pushl
                   $4
26
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| Return Addr of foo |
| EBP of foo |
| 2 |
| |
| |
| |

EBP

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                             $16, %esp
                                                     addl
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                             %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
                                                    Draw the stack frame with FSP and FBP
          subl
                  16(%ebp), %eax
10
                                                                                          EBP
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
13
          addl
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                                      3
                                                     EAX
          pushl
                   %ebp
17
          movl
                   %esp, %ebp
18
                                                     FBX
                                                                     a + b
          pushl
                   %ebx
19
                                                                                           ESP
          subl
                   $4, %esp
20
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                      3
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
24
          subl
                   $4, %esp
25
          pushl
                   $5
          pushl
                   $4
26
```

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller 5 4 3 Return Addr of foo EBP of foo 2

```
call.s
                                           27
                                                      pushl
                                                               $3
                                           28
                                                      call
                                                               bar
1 p:
                                                                                                    Stack Frame of foo's Caller
                                                               $16, %esp
2
                                                      addl
          .zero
                   4
                                           29
                                                      addl
3 bar:
                                           30
                                                               %ebx, %eax
          pushl
                   %ebp
                                           31
                                                      movl
                                                               %eax, p
4
          movl
                   %esp, %ebp
5
                                           32
                                                      nop
6
          subl
                   $16, %esp
                                           33
                                                      movl
                                                               -4(%ebp), %ebx
                   8(%ebp), %edx
                                                      leave
          movl
                                           34
          movl
                   12(%ebp), %eax
                                           35
                                                      ret
8
                                                                                                    Return Addr of foo's Caller
          addl
                   %edx, %eax
9
                                                     Draw the stack frame with FSP and FBP
          subl
                  16(%ebp), %eax
                                                                                                       EBP of foo's Caller
10
                                                                                             EBP
          movl
                  %eax, -4(%ebp)
11
                                                                                                       EBX of foo's Caller
                                                     and contents of EAX, EBX, EDX registers.
          movl
                   -4(%ebp), %eax
12
                   $1, %eax
13
          addl
          leave
14
15
          ret
16 foo:
                                                                        3
                                                       EAX
          pushl
17
                   %ebp
          movl
                   %esp, %ebp
18
                                                                                             ESP
                                                       FBX
                                                                       a + b
                                                                                                       Return Addr of foo
          pushl
                   %ebx
19
          subl
                   $4, %esp
20
21
          movl
                   8(%ebp), %edx
                                                      EDX
                                                                        3
                   12(%ebp), %eax
22
          movl
          leal
                   (%edx,%eax), %ebx
23
24
          subl
                   $4, %esp
25
          pushl
                   $5
          pushl
                   $4
26
```

Padding

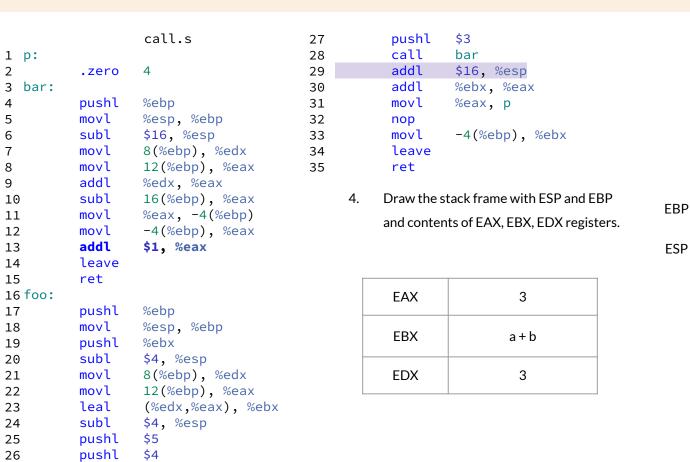
b

а

5

4

3



| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| 5 |
| 4 |
| 3 |
| |
| |
| |
| |
| |
| |

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
                                                              %ebx, %eax
3 bar:
                                          30
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                          34
          movl
                  12(%ebp), %eax
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
                                                    Draw the stack frame with FSP and FBP
          subl
                  16(%ebp), %eax
10
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
          movl
                  -4(%ebp), %eax
12
                   $1, %eax
13
          addl
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                    3+a+b
          pushl
17
                   %ebp
          movl
                   %esp, %ebp
18
                                                     FBX
                                                                     a + b
          pushl
                   %ebx
19
          subl
                   $4, %esp
20
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                       3
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                   $4, %esp
24
          subl
25
          pushl
                   $5
          pushl
                   $4
26
```

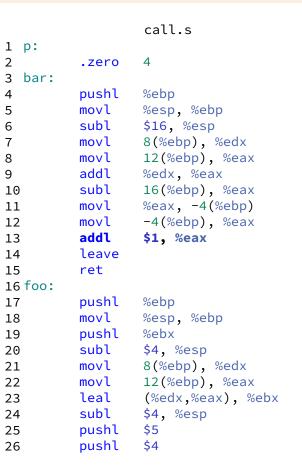
Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller

EBP

```
call.s
                                          27
                                                     pushl
                                                              $3
                                          28
                                                     call
                                                              bar
1 p:
                                                     addl
                                                              $16, %esp
2
          .zero
                   4
                                          29
                                                     addl
3 bar:
                                          30
                                                              %ebx, %eax
          pushl
                   %ebp
                                          31
                                                     movl
                                                              %eax, p
4
          movl
                  %esp, %ebp
5
                                          32
                                                     nop
6
          subl
                  $16, %esp
                                          33
                                                     movl
                                                              -4(%ebp), %ebx
                  8(%ebp), %edx
                                                     leave
          movl
                                          34
                  12(%ebp), %eax
          movl
                                          35
                                                     ret
8
          addl
                  %edx, %eax
9
          subl
                  16(%ebp), %eax
                                                    Draw the stack frame with FSP and FBP
10
          movl
                  %eax, -4(%ebp)
11
                                                    and contents of EAX, EBX, EDX registers.
                  -4(%ebp), %eax
          movl
12
13
          addl
                   $1, %eax
          leave
14
15
          ret
16 foo:
                                                     EAX
                                                                    3+a+b
          pushl
17
                   %ebp
          movl
                  %esp, %ebp
18
                                                     EBX
                                                                EBX of foo's Caller
          pushl
                   %ebx
19
          subl
20
                   $4, %esp
21
          movl
                  8(%ebp), %edx
                                                     EDX
                                                                       3
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                   $4, %esp
24
          subl
25
          pushl
                   $5
          pushl
                   $4
26
```

| Stack Frame of foo's Caller |
|-----------------------------|
| Padding |
| b |
| a |
| Return Addr of foo's Caller |
| EBP of foo's Caller |
| EBX of foo's Caller |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

EBP



```
27
           pushl
                   $3
28
           call
                   bar
                   $16, %esp
           addl
29
           addl
30
                   %ebx, %eax
31
           movl
                   %eax, p
32
           nop
33
           movl
                   -4(%ebp), %ebx
           leave
34
35
           ret
```

 Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 3+a+b |
|-----|---------------------|
| EBX | EBX of foo's Caller |
| EDX | 3 |

EBP

ESP

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller EBP of foo's Caller EBX of foo's Caller

```
call.s
1 p:
2
          .zero
                  4
3 bar:
          pushl
                  %ebp
4
          movl
                  %esp, %ebp
5
6
          subl
                  $16, %esp
                  8(%ebp), %edx
          movl
          movl
                  12(%ebp), %eax
8
          addl
                 %edx, %eax
9
          subl
                 16(%ebp), %eax
10
          movl
                 %eax, -4(%ebp)
11
                 -4(%ebp), %eax
          movl
12
                  $1, %eax
13
          addl
          leave
14
15
          ret
16 foo:
          pushl
                  %ebp
17
          movl
                  %esp, %ebp
18
          pushl
                  %ebx
19
          subl
                  $4, %esp
20
21
          movl
                  8(%ebp), %edx
                  12(%ebp), %eax
22
          movl
          leal
                  (%edx,%eax), %ebx
23
                  $4, %esp
          subl
24
25
          pushl
                  $5
          pushl
                  $4
26
```

```
27
           pushl
                   $3
28
           call
                   bar
                   $16, %esp
           addl
29
           addl
30
                   %ebx, %eax
31
           movl
                   %eax, p
32
           nop
33
           movl
                   -4(%ebp), %ebx
           leave
34
           ret
35
```

Draw the stack frame with ESP and EBP and contents of EAX, EBX, EDX registers.

| EAX | 3+a+b |
|-----|---------------------|
| EBX | EBX of foo's Caller |
| EDX | 3 |

ESP Re

Stack Frame of foo's Caller **Padding** b а Return Addr of foo's Caller