CSO-Recitation 09

CSCI-UA 0201-007

R09: Assessment 08 & Data segment and buffer overflow

Today’s Topics

- Assessment 08
- Data segment and buffer overflow

Assessment 08

Q1 array → Q1.1

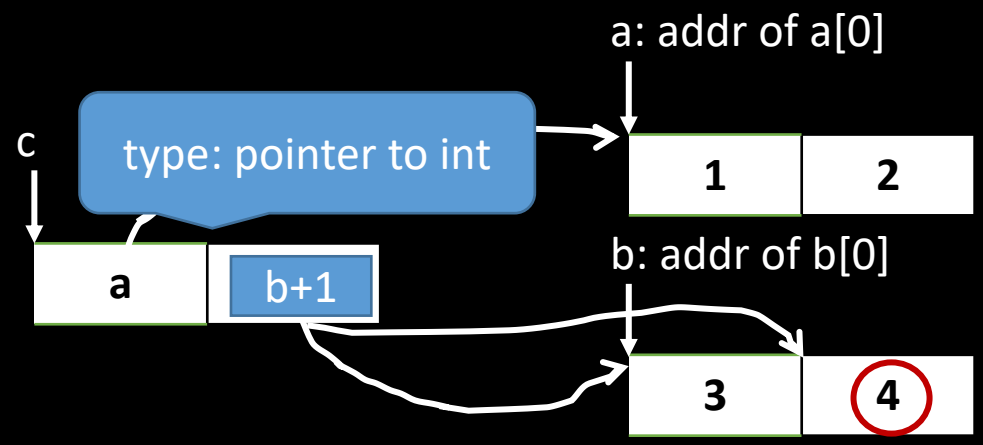
What is the value of *c[1] after executing line 11?

A. 1
B. 2
C. 3
D. 4
E. 5
F. None of the above

Given the following C code

c[1]++;

```
1: voi                    long i)
2: {
3:   arr[i]++;
4: }
5:
6: void test()
7: {
8:  int a[2] = {1, 2};
9:  int b[2] = {3, 4};
10: int *c[2] = {a, b};
11: foo(c, 1);
12:}
```



Q1.2 arr[i]++

If Line 3 is realized using one instruction, what's that

A. `addl $0x8,(%rdi,%rsi,8)`
B. `addl $0x4,(%rdi,%rsi,8)`
C. `addl $0x8,(%rdi,%rsi,4)`
D. `addl $0x4,(%rdi,%rsi,4)`
E. `addq $0x8,(%rdi,%rsi,8)`
F. `addq $0x4, (%rdi,%rsi,8)`
G. `addq $0x8,(%rdi,%rsi,4)`
H. `addq $0x4,(%rdi,%rsi,4)`

Given the following C code

```
1: void foo(int **arr, long i)
2: {
3:   arr[i]++;
4: }
5:
6: void test()
7: {
8:  int a[2] = {1, 2};
9:  int b[2] = {3, 4};
10: int *c[2] = {a, b};
11: foo(c, 1);
12:}
```

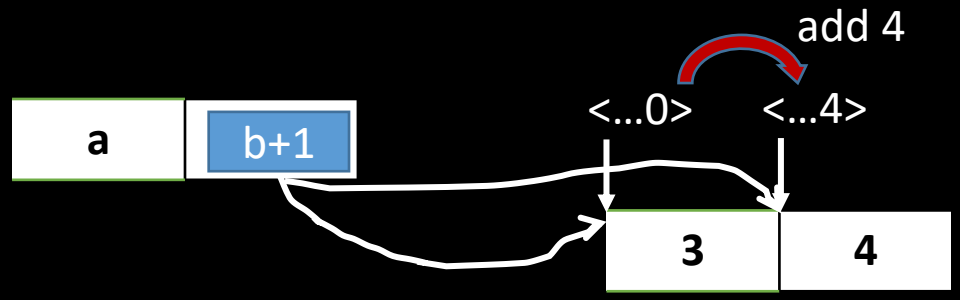
arr[i]:

- Typ… …es => addq
- arr[i]: (%rdi, %rsi, 8)
- addq $…1, (%…rsi, 8)
- addq $0x4 …rdi, %rsi, 8)

start addr of the array

Size of each element (pointer, 8 bytes)

index



Q2 → Q2.1 location of p

Where is the local variable t in test stored?

A. some register
B. memory (data segment)
C. memory (stack)
D. memory (heap)

Given the following C code

```
1: typedef struct kv_pair {
2:   long key;
3:   char* val;
4: } kv_pair;
5:
6: void init_pair(kv_pair *p)
7: {
8:   p->key = -1;
9:   p->val = NULL;
10:}
11:
12: void test()
13: {
14:   kv_pair t;
15:   init_pair(&t);
16:}
```

- local array/struct variables => stack

Q2.2 p->val

char* is 8 byte
=> movq

Given the following C code

```
1: typedef struct kv_pair {
2:   long key;
3:   char* val;
4: } kv_pair;
5:
6: void init_pair(kv_pair *p)
7: {
8:   p->key = -1;
9:   p->val = NULL;
10:}
11:
12: void test()
13: {
14:   kv_pair t;
15:   init_pair(&t);
16:}
```

If Line 9 is realized using one instruction, what is that instruction?

A. `movl $0x0,0x4(%rdi)`
B. `movq $0x0,0x4(%rdi)`
C. `movl $0x0,0x8(%rdi)`
D. `movq $0x0,0x8(%rdi)`
E. `movl $0x0,0x4(%rsi)`
F. `movq $0x0,0x4(%rsi)`
G. `movl $0x0,0x8(%rsi)`
H. `movq $0x0,0x8(%rsi)`

Q3 str_concat

The following C function str_concat appends the src string to the dst string, overwriting the terminating null byte at the end of dst, and then adds a terminating null byte.

- **Q3.1** line 5
- Please fill in the code at line 6 (must be a one liner). *To facilitate automatic grading, please do not have any spaces in your C statement, and make sure to include the end of the statement semicolon*.
- dst[len+i]=src[i];

```
1: void str_concat(char *dst, char *src)
2: {
3:    int len = strlen(dst);
4:    int i;
5:    for (i = 0; src[i]!='\0'; i++) {
6:       ???
7:    }
8:    ???
9: }
```

Q3.2 line 7

Which of the following C statement can be used at line 8 correctly, without compilation nor runtime error?

A. dst[len] = '\0';

B. dst[len] = NULL;

C. dst[len] = 0;

D. dst[len+i] = '\0';

E. dst[len+i] = NULL;

F. dst[len+i] = 0;

NULL: means a null pointer.
In C language, NULL is defined as (void*)0.
E triggers compilation warning

- makes integer from pointer without a cast [enabled by default]

A bit vague, so select E or not, we give full mark

Q4. Given the following C program (which invokes str_concat defined in Q3)

```c
void dangerous()
{
  char buf1[8] = "hello";
  char buf2[8] = "world";
  str_concat(buf1, buf2);
}

int main()
{
  dangerous();
  printf("i wonder if this program is correct\n");
}
```

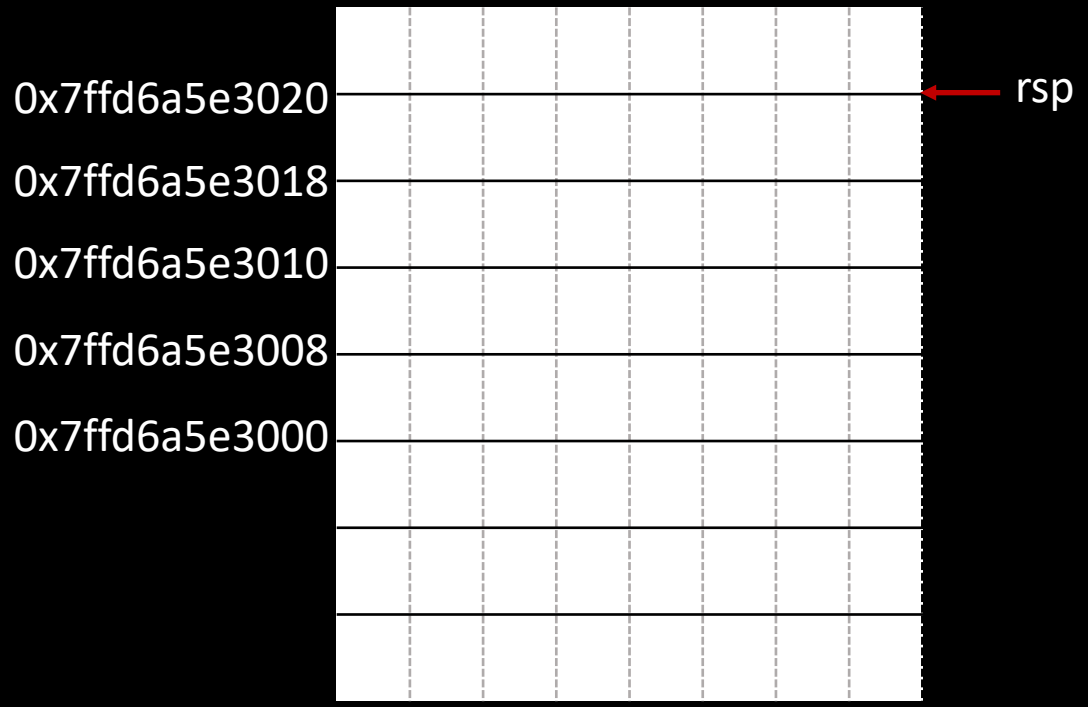
Suppose the following assembly is generated for the above C program (including str_concat). Please assume that the addresses to the left of each instruction shown below are the actual addresses where the instructions are stored at during runtime.

```
000000000550068a <str_concat>:
             68a:       48 83 ec 28             subq    $0x18,%rsp
             ...omitted....
             711:       c3                      ret

00000000055006ba <dangerous>:
             6ba:       48 83 ec 10             subq    $0x10,%rsp
             6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
             6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
             6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
             6d7:       48 89 04 24             movq    %rax,(%rsp)
             6db:       48 89 e6                movq    %rsp,%rsi
             6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
             6e3:       e8 82 ff ff ff          callq   66a <str_concat>
             6e8:       48 83 c4 10             addq    $0x10,%rsp
             6ec:       c3                      ret

00000000055006ed <main>:
             6ed:       48 83 ec 08             subq    $0x8,%rsp
             6f1:       b8 00 00 00 00          movq    $0x0,%eax
             6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
             6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
             702:       bf 01 00 00 00          movq    $0x1,%edi
             707:       b8 00 00 00 00          movq    $0x0,%eax
             70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
             711:       b8 00 00 00 00          movq    $0x0,%eax
             716:       48 83 c4 08             addq    $0x8,%rsp
             71a:       c3                      ret
```

Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

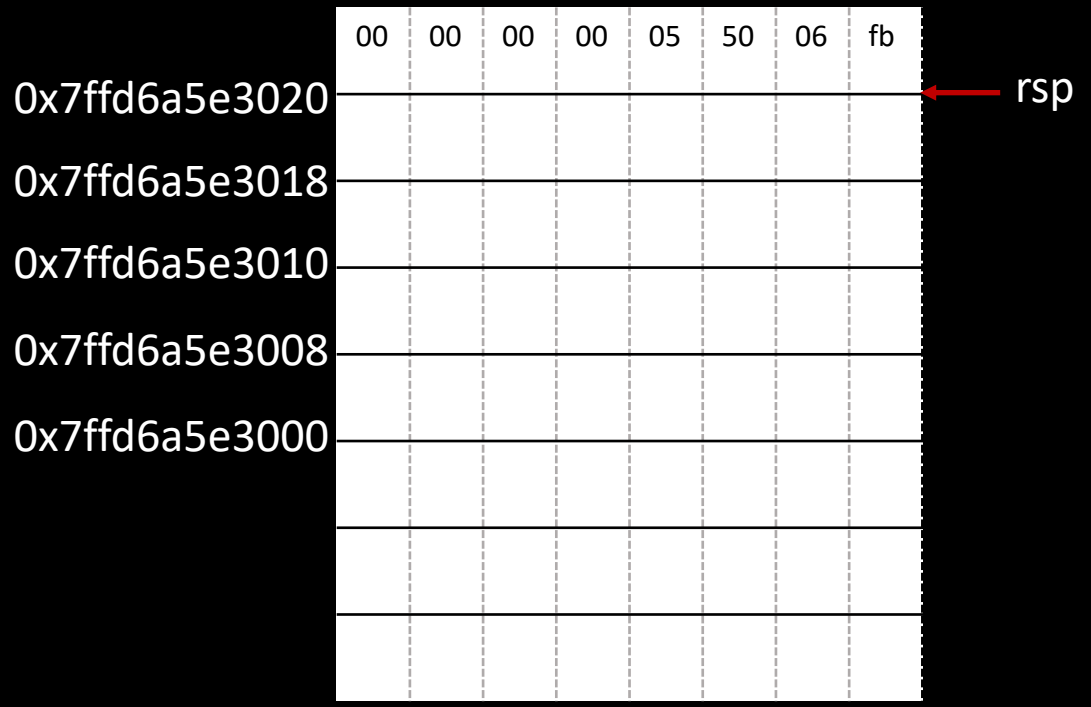
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

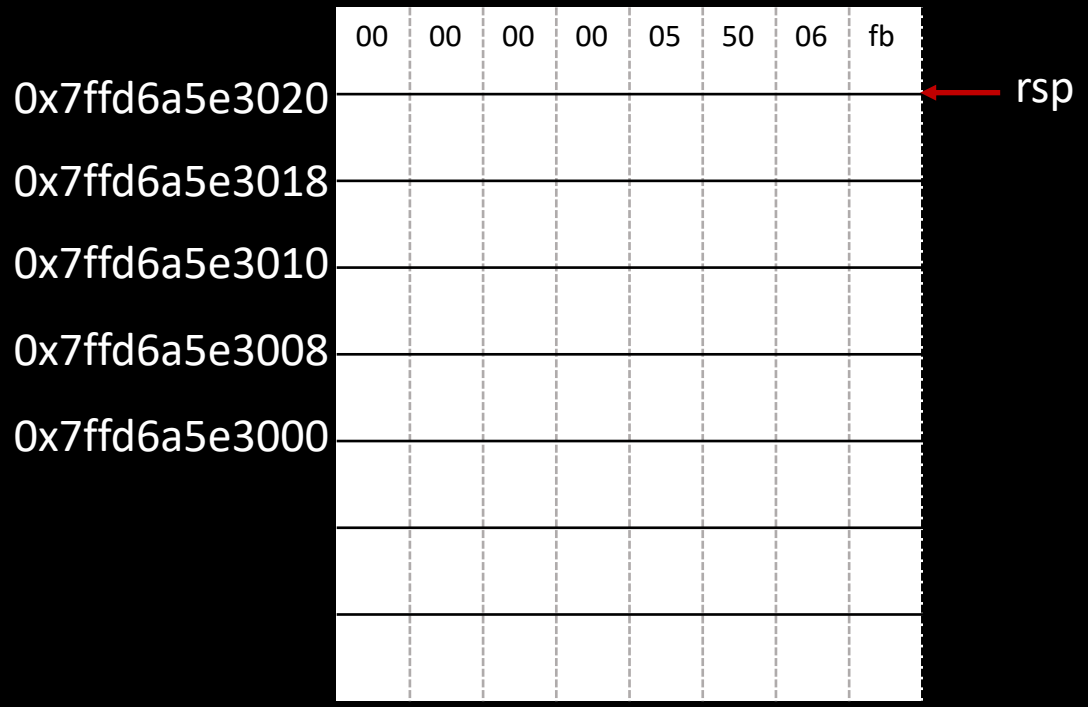
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

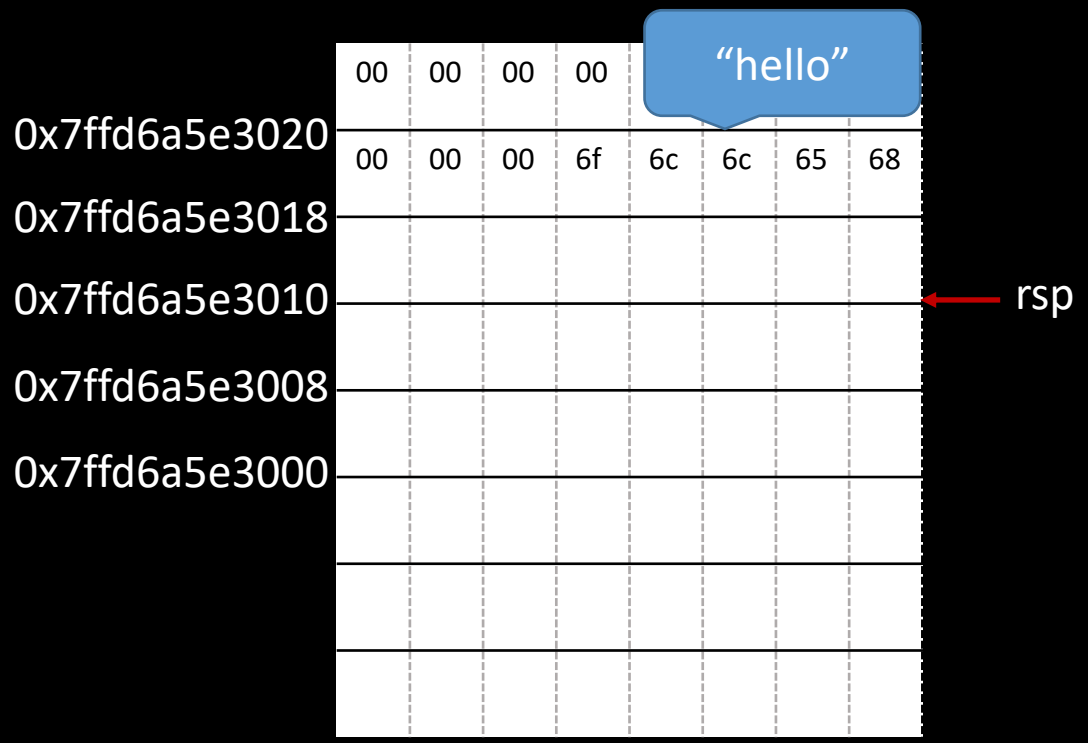
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

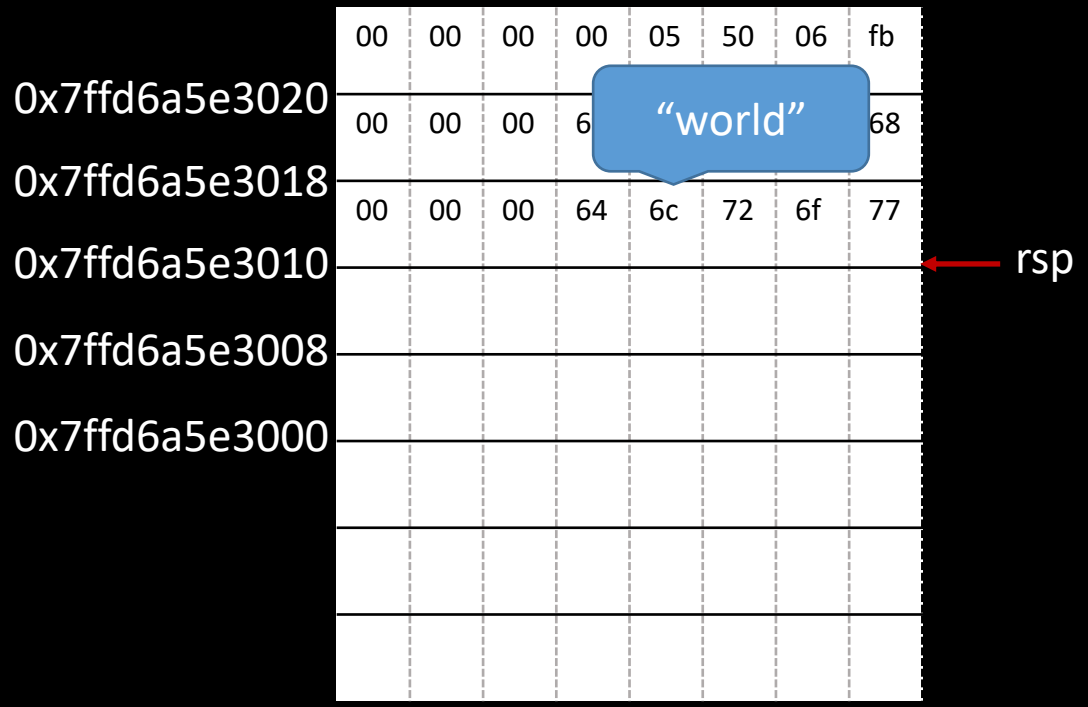
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

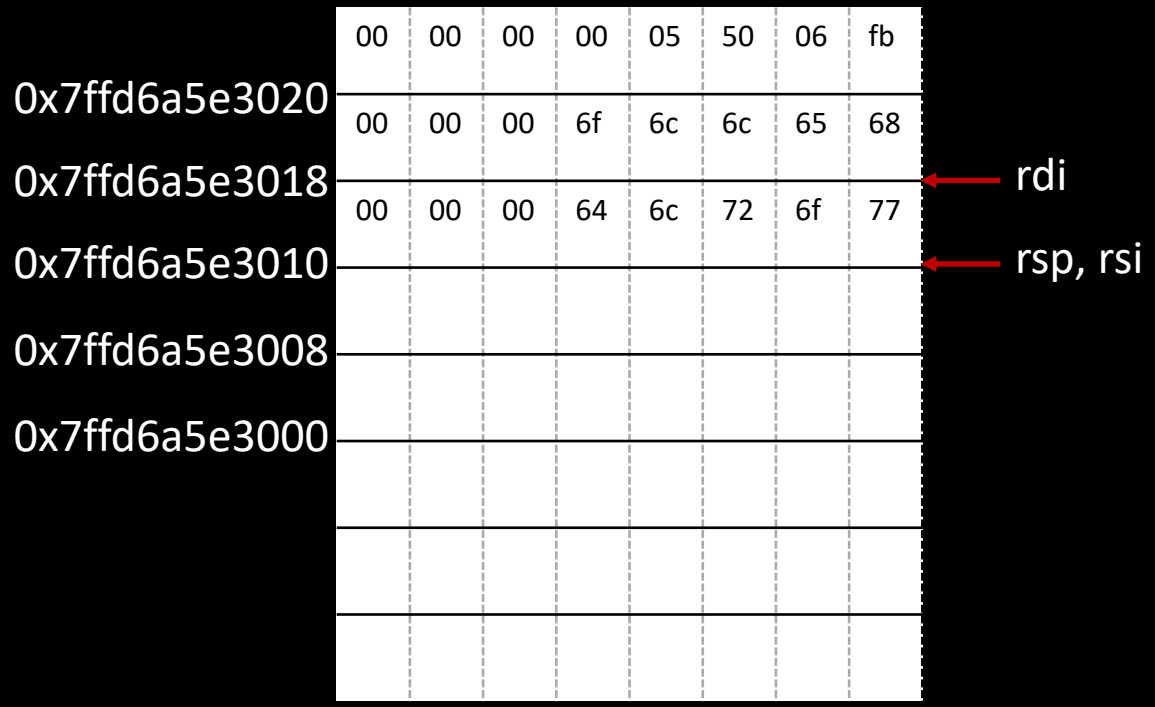
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

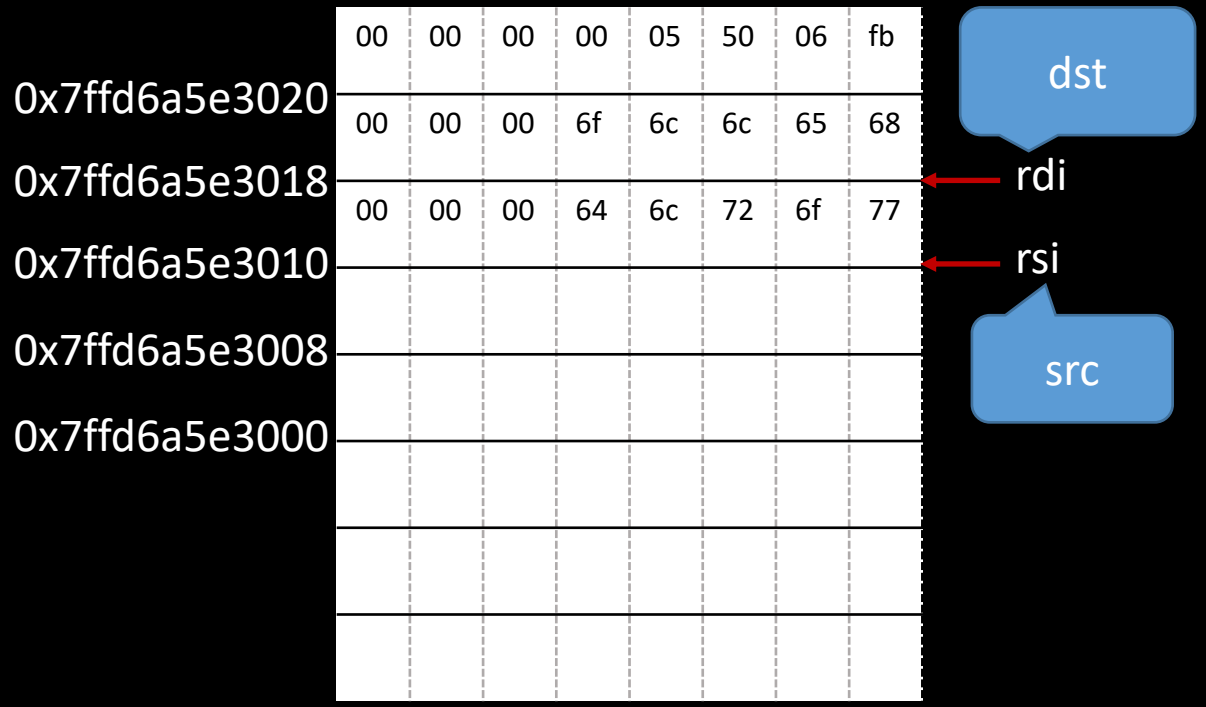
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



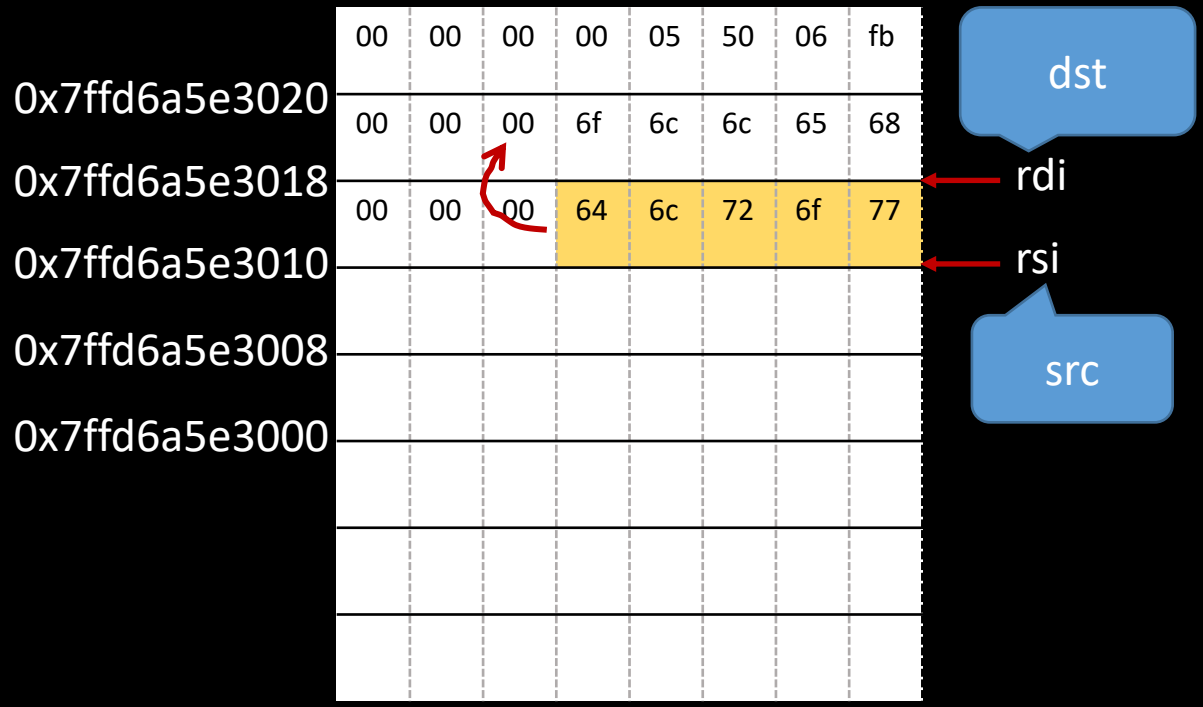
appends the src string to the dst string, overwriting the terminating null byte at the end of dst, and then adds a terminating null byte.

```
1: void str_concat(char *dst, char *src)
2: {
3:    int len = strlen(dst);
4:    int i;
5:    for (i = 0; src[i]!='\0'; i++) {
6:       ???
7:    }
8:    ???
9: }
```

Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



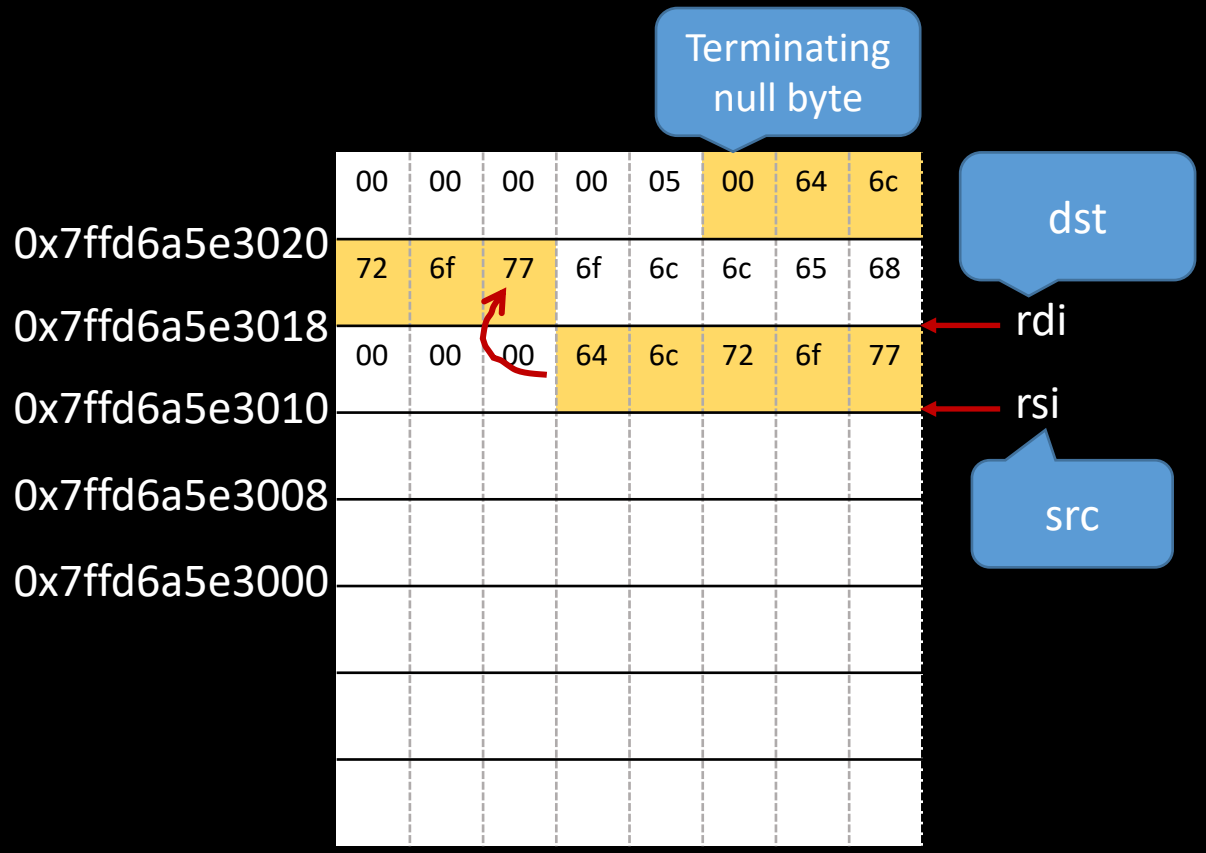
appends the src string to the dst string, overwriting the terminating null byte at the end of dst, and then adds a terminating null byte.

```
1: void str_concat(char *dst, char *src)
2: {
3:    int len = strlen(dst);
4:    int i;
5:    for (i = 0; src[i]!='\0'; i++) {
6:       ???
7:    }
8:    ???
9: }
```

Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



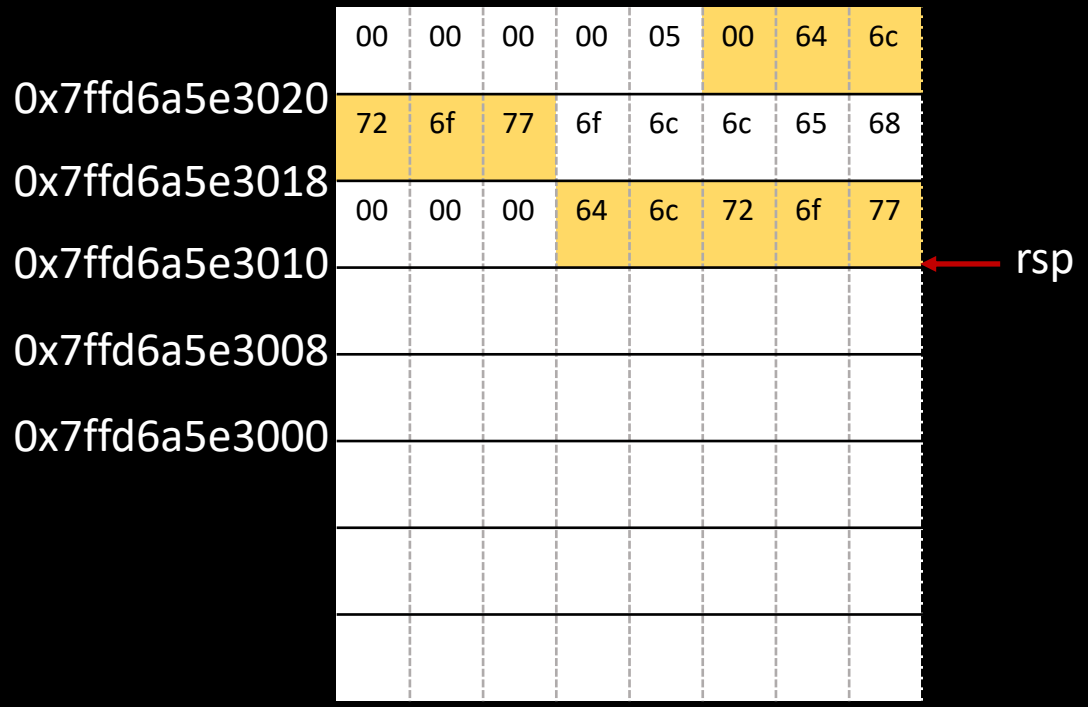
appends the src string to the dst string, overwriting the terminating null byte at the end of dst, and then adds a terminating null byte.

```
1: void str_concat(char *dst, char *src)
2: {
3:    int len = strlen(dst);
4:    int i;
5:    for (i = 0; src[i]!='\0'; i++) {
6:       ???
7:    }
8:    ???
9: }
```

Q4

Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

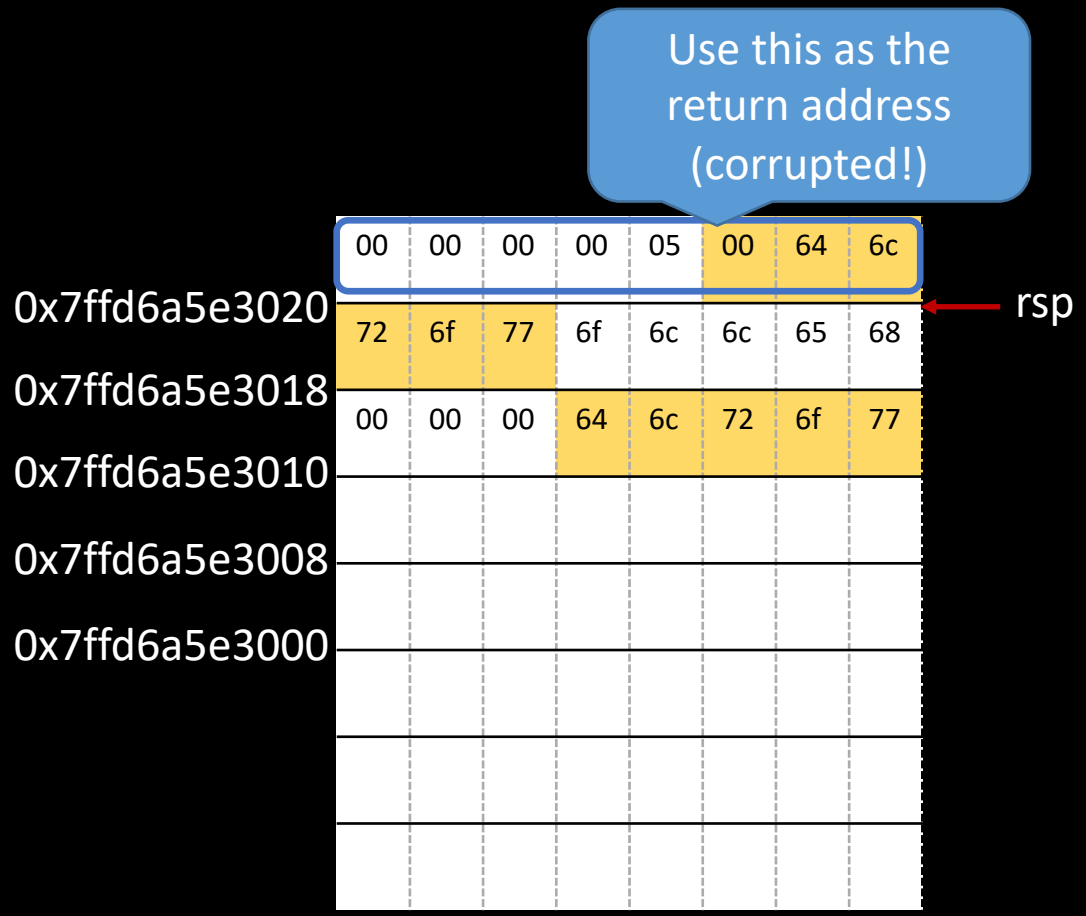
```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



Q4

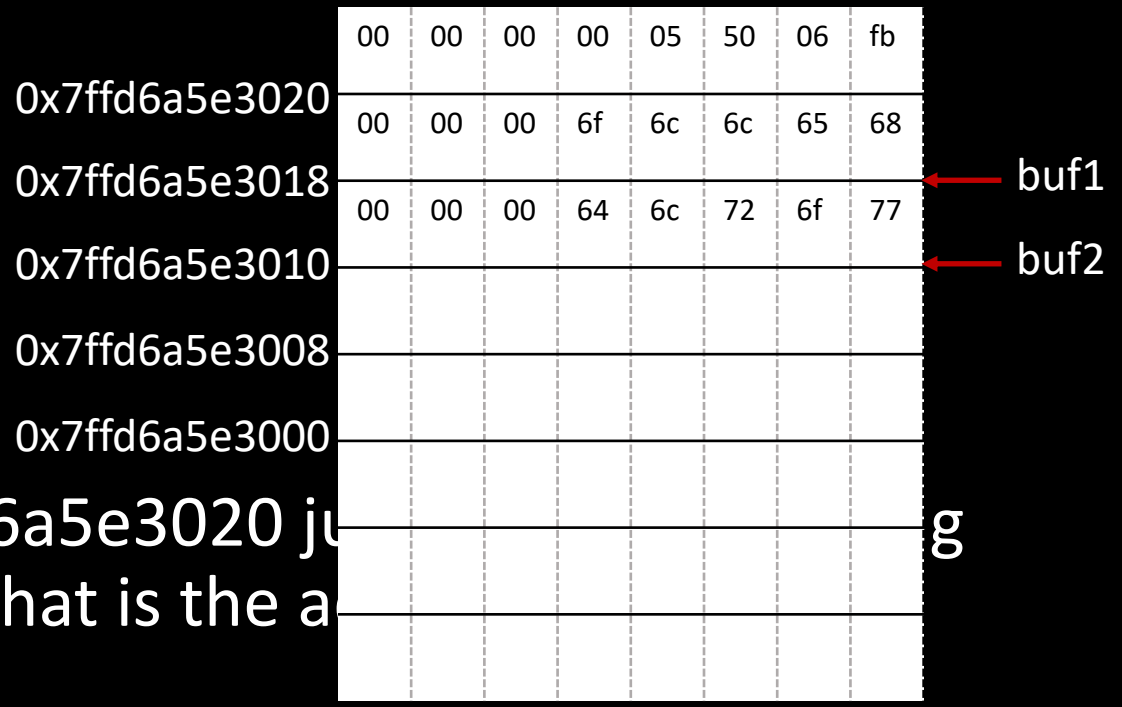
Suppose the value of %rsp is 0x7ffd6a5e3020 just prior to executing the first instruction of dangerous

```
000000000550068a <str_concat>:
            68a:       48 83 ec 28             subq    $0x18,%rsp
            ...omitted....
            711:       c3                      ret

00000000055006ba <dangerous>:
            6ba:       48 83 ec 10             subq    $0x10,%rsp
            6be:       48 b8 68 65 6c 6c 6f    movq    $0x6f6c6c6568,%rax
            6c8:       48 89 44 24 08          movq    %rax,0x8(%rsp)
            6cd:       48 b8 77 6f 72 6c 64    movq    $0x646c726f77,%rax
            6d7:       48 89 04 24             movq    %rax,(%rsp)
            6db:       48 89 e6                movq    %rsp,%rsi
            6de:       48 8d 7c 24 08          leaq    0x8(%rsp),%rdi
            6e3:       e8 82 ff ff ff          callq   66a <str_concat>
            6e8:       48 83 c4 10             addq    $0x10,%rsp
            6ec:       c3                      ret

00000000055006ed <main>:
            6ed:       48 83 ec 08             subq    $0x8,%rsp
            6f1:       b8 00 00 00 00          movq    $0x0,%eax
            6f6:       e8 bf ff ff ff          callq   6ba <dangerous>
            6fb:       48 8d 35 a2 00 00 00    leaq    0xa2(%rip),%rsi
            702:       bf 01 00 00 00          movq    $0x1,%edi
            707:       b8 00 00 00 00          movq    $0x0,%eax
            70c:       e8 2f fe ff ff          callq   540 <__printf_chk@plt>
            711:       b8 00 00 00 00          movq    $0x0,%eax
            716:       48 83 c4 08             addq    $0x8,%rsp
            71a:       c3                      ret
```



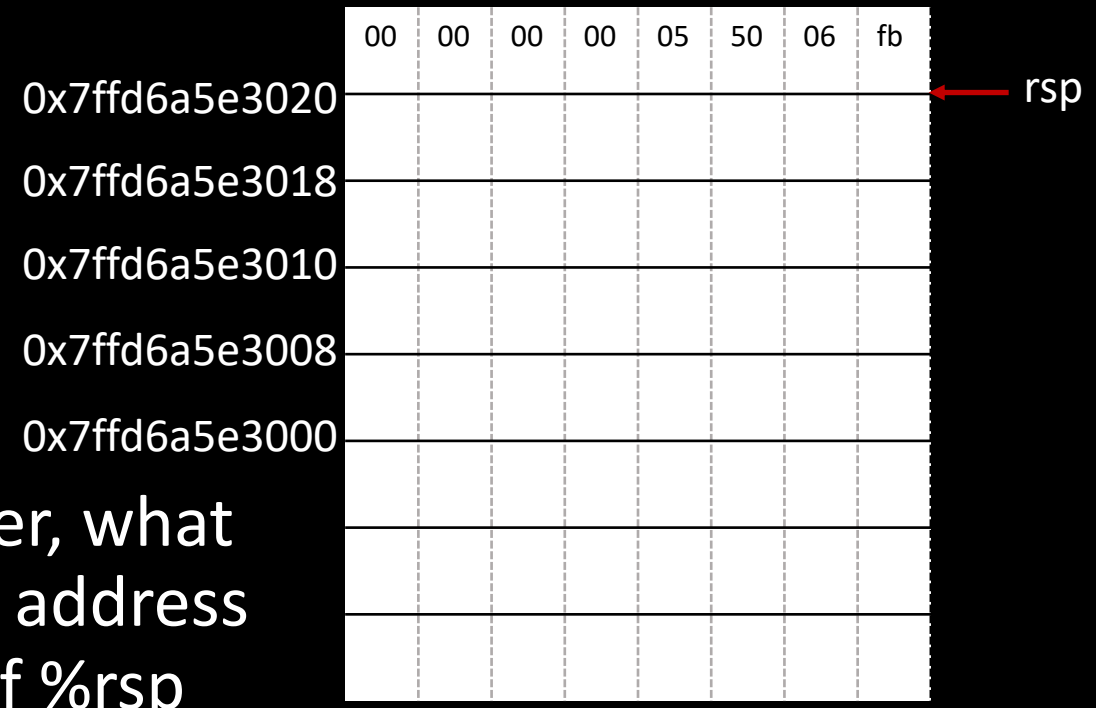
Q4

```
void dangerous()
{
  char buf1[8] = "hello";
  char buf2[8] = "world";
  str_concat(buf1, buf2);
}

int main()
{
  dangerous();
  printf("i wonder if this program is correct\n");
}
```



- **Q4.1** &buf1[0]
- Suppose the value of %rsp is 0x7ffd6a5e3020 ju g the first instruction of dangerous, what is the a element of buf1 (aka &buf1[0])?
- 0x7ffd6a5e3018
- **Q4.2** &buf2[0]
- Using the same premise of Q4.1 earlier, what is the address of the first element of buf2 (aka &buf2[0])?
- 0x7ffd6a5e3010

Q4



- **Q4.3**
- Using the same premise of Q4.1 earlier, what are the 8 bytes stored in the memory address 0x7ffd6a5e3020 (which is the value of %rsp just prior to executing the first instruction of dangerous)?
- 0x55006fb

Q4

- **Q4.4** Which of the following statements are true?

A. This program has no buffer overflow bugs and will execute correctly.

B. This program has a buffer overflow bug, but it will nevertheless execute without a problem because the compiler has protected the stack using a canary.

C. This program has a buffer overflow bug, but it will nevertheless execute without a problem because the compiler has allocated extra space on the stack that cushions the overflow.

D. This program has a buffer overflow bug which is likely to manifest as a segmentation fault.

E. buf1 is overflown during execution.

F. buf2 is overflown during execution.

Q4

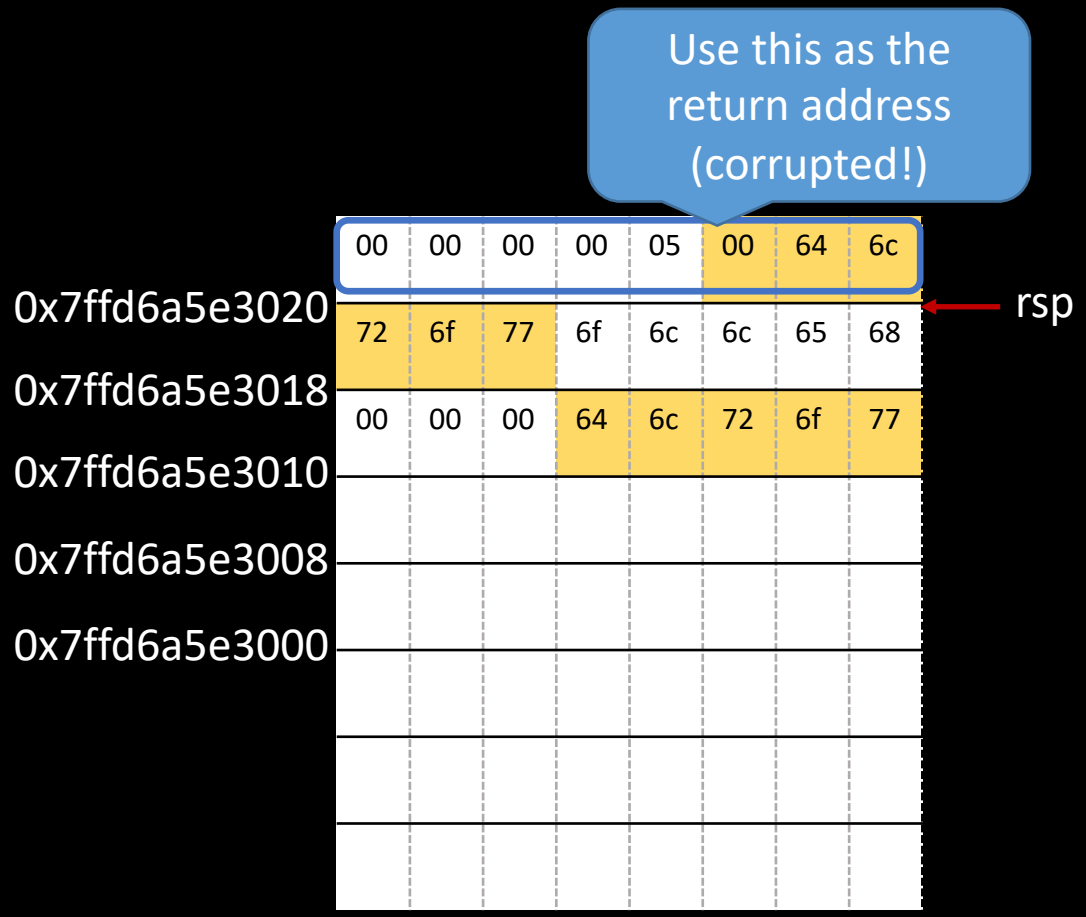
- **Q4.5** last instruction

If running this program results in a segmentation fault. What is the last instruction executed before the segmentation fault occurs?

A. The ret instruction in main function
B. The ret instruction in dangerous function
C. The ret instruction in str_concat function
D. The instruction to deallocate stack in dangerous, i.e. addq $0x10,%rsp.

Q4

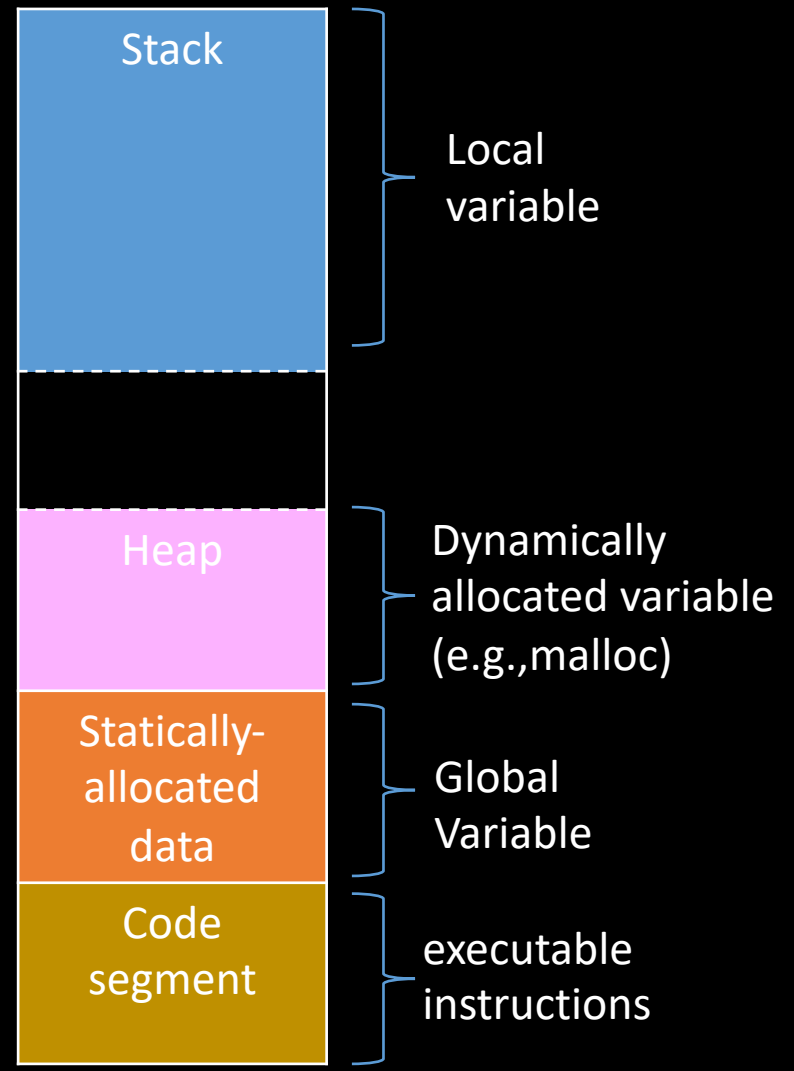
- **Q4.6**
- If running this program results in a segmentation fault, what is the memory address that corresponds to the illegal memory access? You should assume the same premise as Q4.1.
- 0x500646c



Data segment

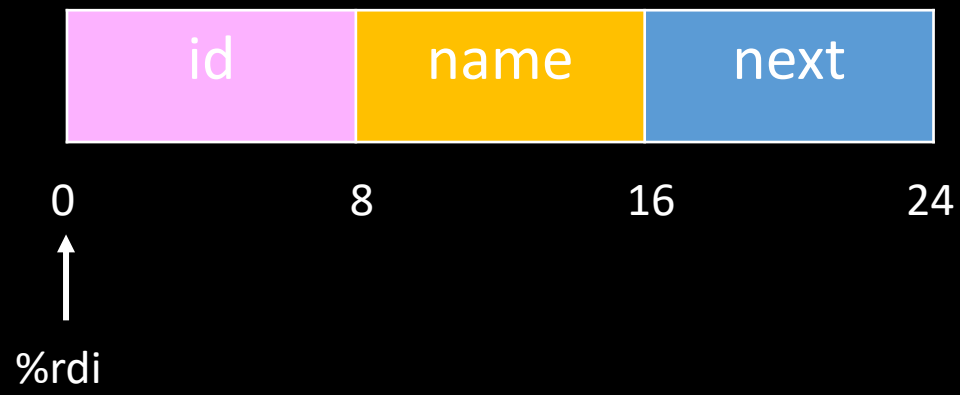
Data segment

- Local variables
  - char, int, long, … (primitive data types) and pointers => use registers whenever possible
  - stack otherwise
  - local array/struct variables => stack



Example of Array/Struct accessing

```
typedef struct node {
        long id;
        char *name;
        struct node *next;
}node;
```




```
                    %rdi    %rsi          %rdx
void init_node(node*n, long id, char *name){
        n->id=id;
        n->name=name;
        n->next=NULL;
}
```

```
movq %rsi, (%rdi)
movq %rdx, 8(%rdi)
movq $0, 16(%rdi)
```

Buffer Overflow

Not all buggy memory references access “illegal” memory

Buffer Overflow

- Have learnt about the memory layout
- If an instruction tries to access some invalid memory
  - Segmentation fault occurs
- But not all buggy memory references access “illegal” memory
  - Buffer overflow exploits

Buffer Overflow

- Buffer overflow on the stack
- Buffer overflow overwrites the return address
  - attackers may carefully chosen return address, executes malicious code
  - code injection buffer overflow attacks

Defenses

- Write correct code to avoid overflow vulnerability
  - Use safe APIs to limit buffer lengths

```
char *
strcpy(ch
{
    char *save = t
    for (; (*dst = *src)!= '\0'; ++from, ++to);
    return(save);
}
```

Copy src to dst until the end of src.
When length(src) > sizeof(dst), overflow!

Defenses

- Write correct code to avoid overflow vulnerability
  - Use safe APIs to limit buffer lengths

```
char *
strcpy(ch
{
    char *save = t
    for (; (*dst = *src)!= '\0'; ++from, ++to);
    return(save);
}
```

Copy src to dst until the end of src. When length(src) > sizeof(dst), overflow!

Copy src to dst with nlen chars

```
char *strncpy(char* dst, const char* src, int nlen);
```

```
char *src = ....
char dst[100];
strncpy(dst, src, sizeof(dst));
```

Limit the size, and thus it wouldn't overflow.

Defenses

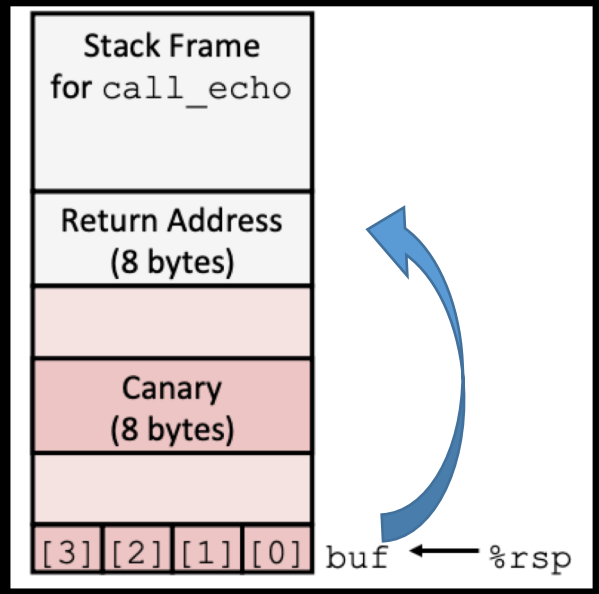
- Write correct code to avoid overflow vulnerability
  - Use safe APIs to limit buffer lengths
  - Use a memory-safe language
    - E.g., python checks every array access and raise exception (crash the program) if the access is beyond the bound

```
a = [0, 1, 2] # python array of size 3
print(a[10])
```

10 >= 3
This throws an IndexError exception!

Defenses

- Mitigate attacks despite buggy code
  - will be an always on-going project, attack and defense itself are alternately developed
    - Security research domain
  - One idea to prevent control flow hijiacking: catch over-written return address before invocation
    - place special value (“canary”) just beyond buffer
    - GCC implementation: -fstack-protector



Simple buffer overflow will overwrite canary in order to reach return address

Backup: Compiler optimization

Tries to minimize or maximize some attributes of an executable computer program

Compiler Optimization

- Goal: generate efficient, correct machine code
  - generally implemented using a sequence of *optimizing transformation*

- Common optimization
  - code motion
  - use simpler instructions
  - reuse common subexpressions

Examples

- Code motion

```
void set_arr(long *arr, long n) {
  for (long i = 0; i < n; i++)
    arr[i] = n*n;
}
```

```
void set_arr(long *arr, long n) {
  long t = n*n;
  for (long i = 0; i < n; i++)
    arr[i] = t;
}
```

- Use simpler instruction:
  - return 9*x

```
// %rdx stores x
mov %rdx,%rax
shl $0x3,%rax
add %rdx,%rax
```

```
// %rax stores x
lea (%rax,%rax,8),%rax
```

8*%rax+%rax=9*%rax
3 instructions => 1 instruction

Optimization -- GCC

- gcc’s optimization levels: -O, -O2, -O3, -Og, -O0, -Os, -Ofast
- learn more here: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Optimization -- GCC

- Tip: when debugging your code, it may help to disable optimizations
  - Replace –Og (if any) with –O0 in the Makefile CFLAGS

- To learn more about GCC's optimizations, invoke GCC with -Q --help=optimizers to find out the exact set of optimizations that are enabled at each level