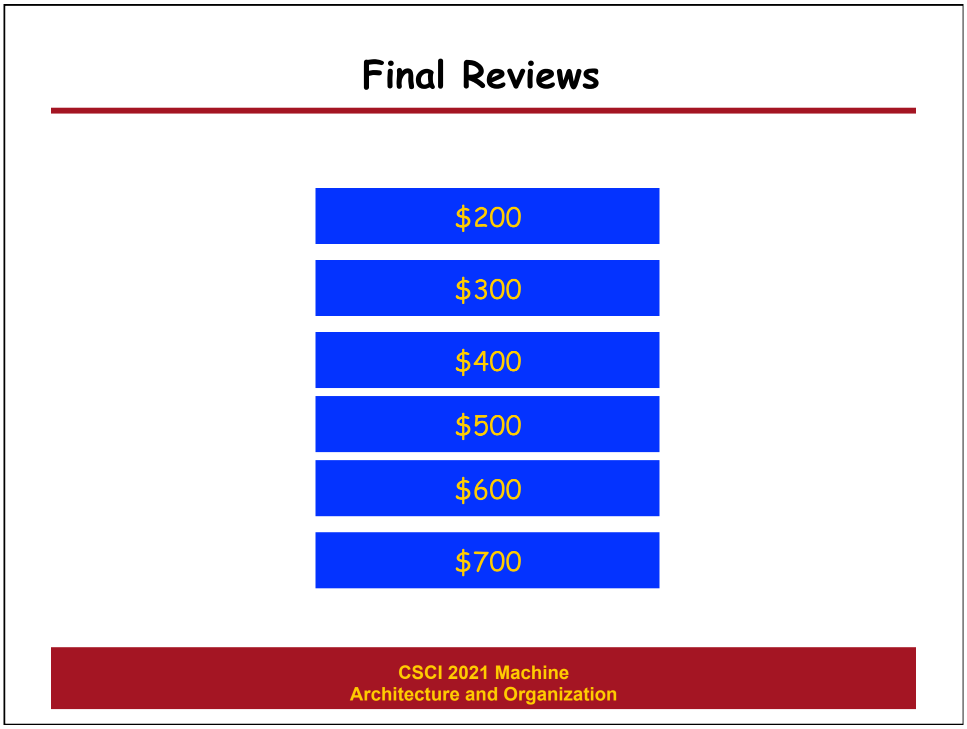
Final Review

CSCI 2021: Machine Architecture and Organization

Antonia Zhai
Department Computer Science and Engineering
University of Minnesota
http://www.cs.umn.edu/~zhai

University of Minnesota

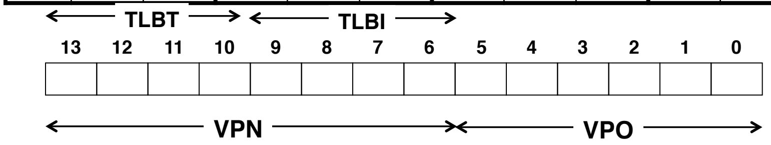


Final Reviews

- $200
- $300
- $400
- $500
- $600
- $700

$200

Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
3	–	0	9	0D	1	0	0A	0	7	02	0
3	2D	1	2	–	0	4	–	0	A	–	0
0	0A	0	8	–	0	6	–	0	3	–	0
7	–	0	0	0D	0	A	34	0	2	–	0



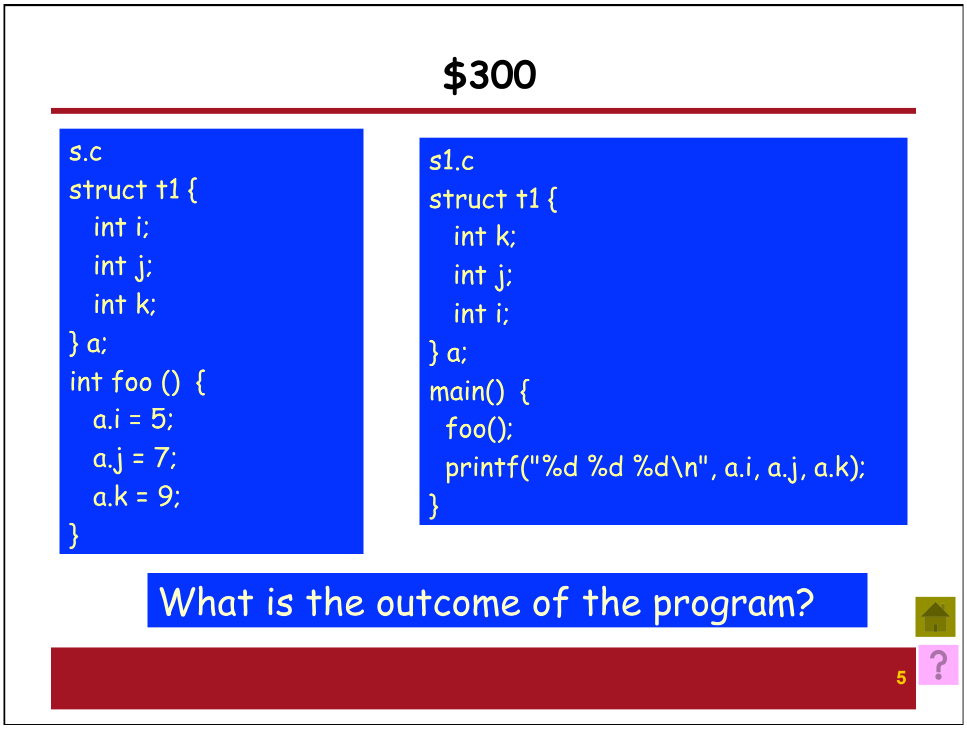
Find out the physical address of the Following virtual address:

0x021E

Answer: $200

Virtual Address = 0x0001000011110

Physical address = Cannot be translated

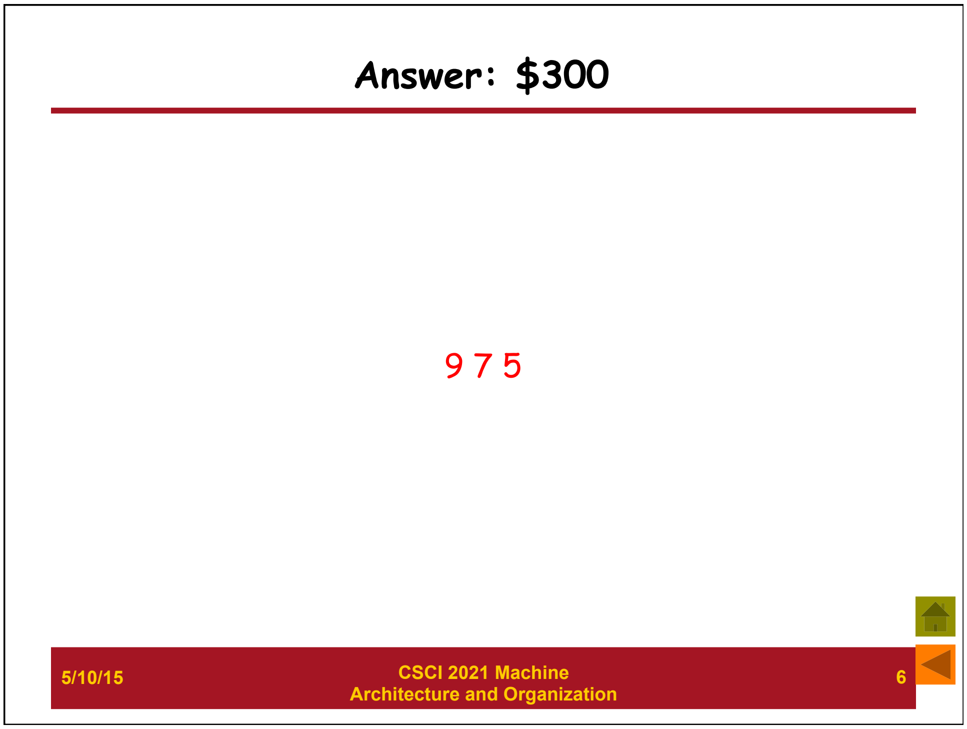


$300

```
s.c
struct t1 {
   int i;
   int j;
   int k;
} a;
int foo () {
   a.i = 5;
   a.j = 7;
   a.k = 9;
}
```

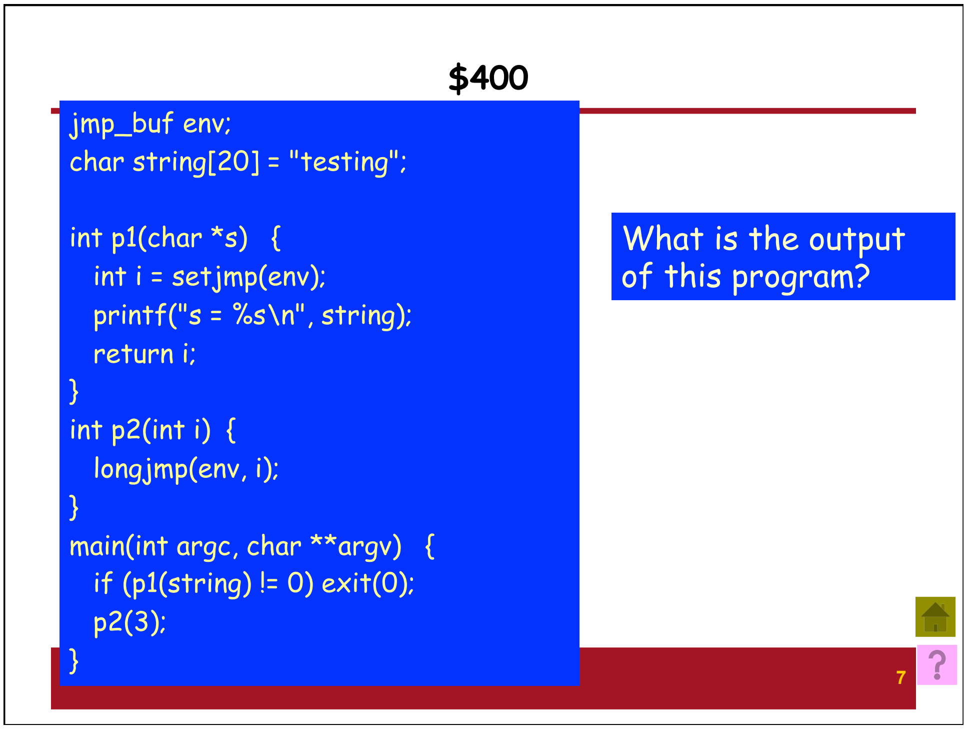
```
s1.c
struct t1 {
   int k;
   int j;
   int i;
} a;
main() {
  foo();
  printf("%d %d %d\n", a.i, a.j, a.k);
}
```

What is the outcome of the program?



Answer: $300

9 7 5

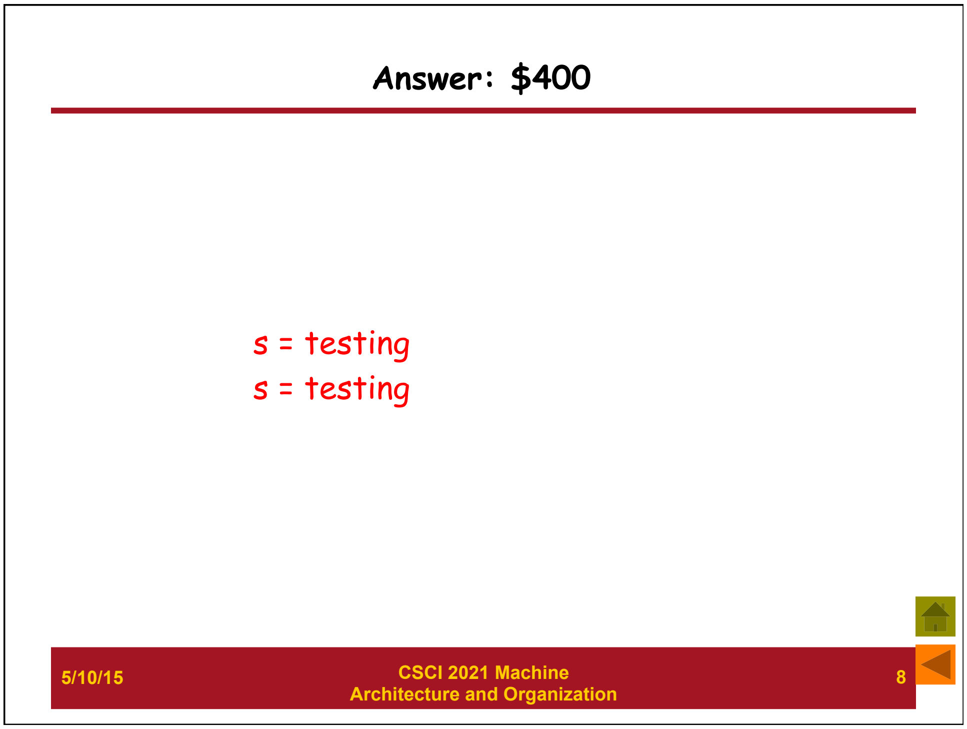


$400

```
jmp_buf env;
char string[20] = "testing";

int p1(char *s)  {
   int i = setjmp(env);
   printf("s = %s\n", string);
   return i;
}
int p2(int i) {
   longjmp(env, i);
}
main(int argc, char **argv)  {
   if (p1(string) != 0) exit(0);
   p2(3);
}
```

What is the output of this program?



Answer: $400

```
s = testing
s = testing
```

$500

Memory blocks are 8-byte aligned, thus, only the 29 higher order bits in the header and footer are needed to record block size. The remaining bits are used as:

Bit 0: whether the current block is free

Bit 1: whether the the previous adjacent block is free

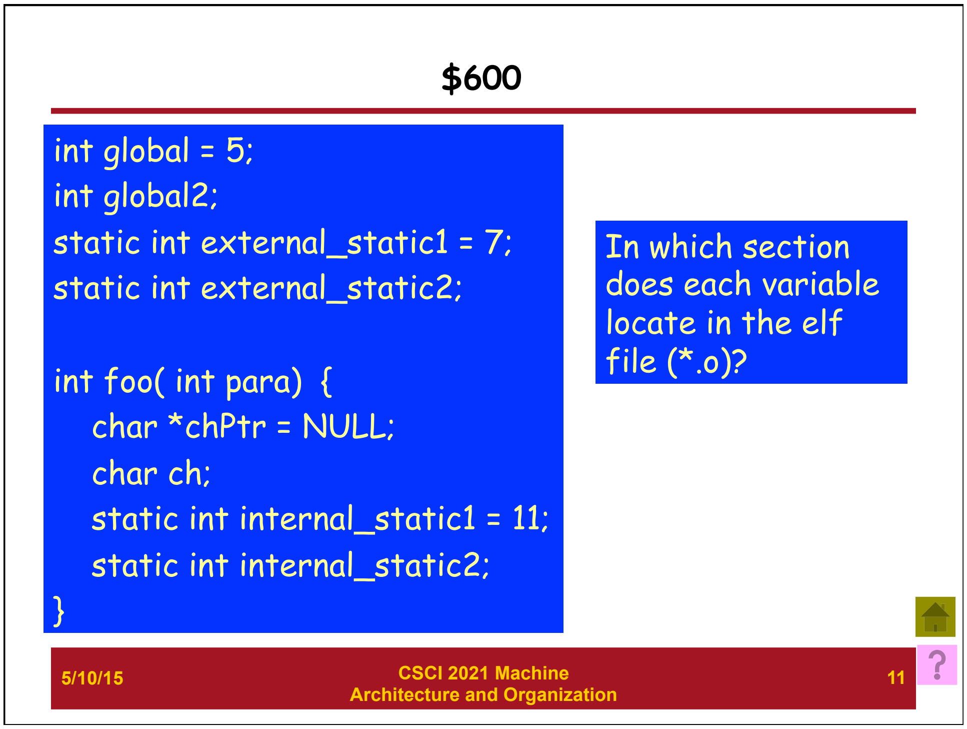
Bit 2: always set to 0

What is the memory content after free(0x400b010)

Address	**0x400b028**	0x400b024	0x400b020	0x400b01c	0x400b018	0x400b014	0x400b010	0x400b00c	0x400b008	0x400b004	0x400b000	0x400affc
Before	0x00000012	0x400b611c	0x400b512c	0x00000012	0x00000013	0x400b511c	0x400b601c	0x00000013	0x00000013	0x400b601c	0x400b511c	0x00000013
After		0x400b024	0x400b512c			0x400b511c	0x400b601c			0x400b601c	0x400b511c	

Answer: $500

Address	**0x400b028**	0x400b024	0x400b020	0x400b01c	0x400b018	0x400b014	0x400b010	0x400b00c	0x400b008	0x400b004	0x400b000	0x400affc
Before	0x00000012	0x400b611c	0x400b512c	0x00000012	0x00000013	0x400b511c	0x400b601c	0x00000013	0x00000013	0x400b601c	0x400b511c	0x00000013
After	0x00000022	0x400b024	0x400b512c	0x00000012	0x00000013	0x400b511c	0x400b601c	0x00000022	0x00000013	0x400b601c	0x400b511c	0x00000013

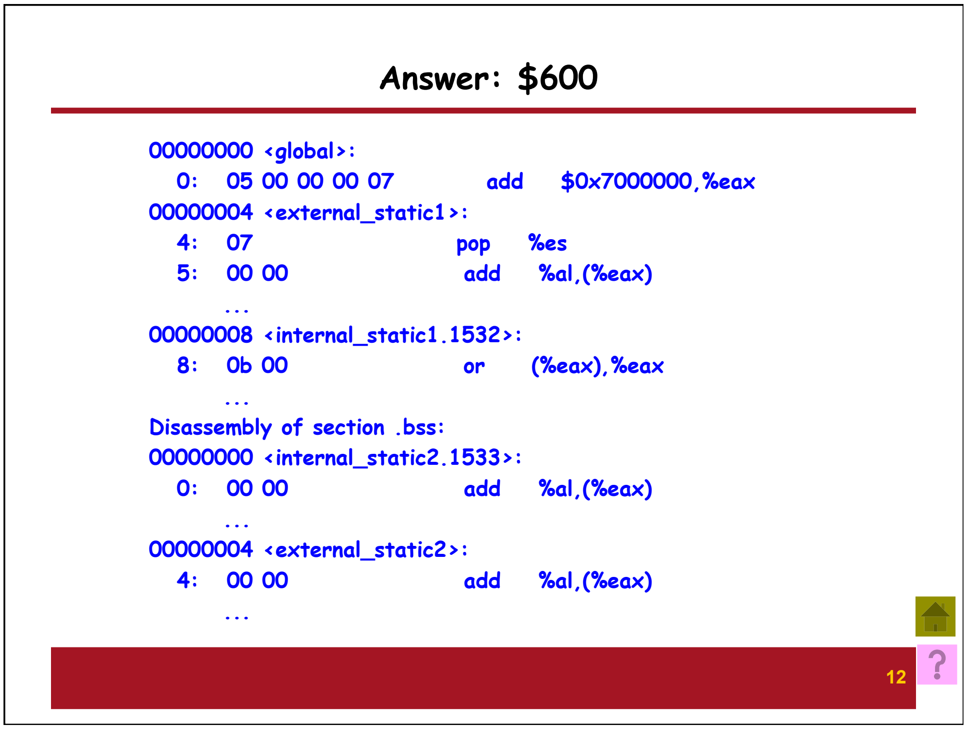


$600

```
int global = 5;
int global2;
static int external_static1 = 7;
static int external_static2;

int foo( int para) {
    char *chPtr = NULL;
    char ch;
    static int internal_static1 = 11;
    static int internal_static2;
}
```

In which section does each variable locate in the elf file (*.o)?



Answer: $600

```
00000000 <global>:
   0:   05 00 00 00 07          add    $0x7000000,%eax
00000004 <external_static1>:
   4:   07                     pop    %es
   5:   00 00                   add    %al,(%eax)
        ...
00000008 <internal_static1.1532>:
   8:   0b 00                   or     (%eax),%eax
        ...
Disassembly of section .bss:
00000000 <internal_static2.1533>:
   0:   00 00                   add    %al,(%eax)
        ...
00000004 <external_static2>:
   4:   00 00                   add    %al,(%eax)
        ...
```

$700

```
void foo()  {
    int a[3];
    char buf[4];
    a[0] = 0xF0;
    gets(buf);
    return;
}
```

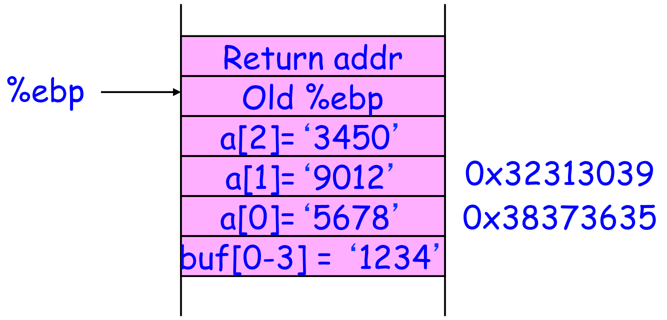
Character ‘1’ correcponds to hex number 0x31

```
foo:  pushl   %ebp
      movl    %esp, %ebp
      subl    $36, %esp
      movl    $0xF0, -12(%ebp)
      leal    -16(%ebp), %eax
      movl    %eax, (%esp)
      call    gets
      …
```

If the user entered “123456789012345”, before foo() returns, what is the value of the following references:

a[0], a[1], a[2], a[3], a[4]

Answer: $700


%ebp
Return addr
Old %ebp
a[2]= ‘3450’
a[1]= ‘9012’
a[0]= ‘5678’
buf[0-3] = ‘1234’
0x32313039
0x38373635