

# CSAPP 某年期末

## 判断题 2pts \* 8=16pts

1-1  $(x|-x) \gg 31 == -1$  x 的类型是 int

1-2 在 UNIX 系统中，文件就是字节序列，每个 I/O 设备都可以看成文件。  一切皆文件

1-3  $-1 < 0U$

1-4 cmpq (%rax), 6(%rsp)

1-5 fork 调用一次，返回两次

1-6  $x \gg 3 == x/8$

1-7 连接器的两个主要任务是符号解析和地址重定位

1-8 lea %r12, 0x2006ae(%rip)

## 单选题 2pts \* 20 = 40pts

C 2-1 运行 C 代码：

```
short si = -128;
unsigned short usi = si;
```

那么 usi 的值是：

- A. 65407
- B. 128
- C. 65408
- D. 127

B 2-2 对于下面两个程序

```
#include <stdio.h>
void f(void);
int x = 15213;
int main() {
    f();
    printf("x=%d\n", x);
    return 0;
}
```

```
int x;
void f() {
    x = 15212;
}
```

首先执行命令 `gcc -o foo a1.c a2.c`，接着执行 `foo` 后，x 的输出结果是

- A. 15213
- B. 15212
- 垃圾选项

A 2-3 对于某8位计算机，整数用补码表示，假设  $x=60$ ,  $y=-30$ ，则  $x+y$  的机器数以及相对应的 OF (1为溢出, 0为不溢出) 为：

- A. 0x1E, 0
- B. 0xE2, 1
- C. 0x1E, 1
- D. 0xE2, 0

D 2-4 对于 gcc 工具链而言，cpp、cc1、as、ld 分别代表着：

- A. 编译器、预处理器、链接器、汇编器

- B. 编译器、预处理器、汇编器、链接器
- C. 预处理器、链接器、编译器、汇编器
- D. 预处理器、编译器、汇编器、链接器

A 2-5 已知变量 y 的类型是 float, 位于地址 0x400 处, 它的十六进制为 0x01234567, 在采用小端模式存储的机器中, 地址范围为 0x400 ~ 0x403 处存储的字节依次为:

- A. 0x67 0x45 0x23 0x01
- B. 0x01 0x23 0x45 0x67
- 垃圾选项

A 2-6 十六位补码整数的范围

$$W = 16$$

- A.  $-2^{15} \sim + (2^{15} - 1)$
- B.  $-(2^{15} - 1) \sim + (2^{15} - 1)$
- C.  $-2^{16} \sim + (2^{16} - 1)$
- D.  $-(2^{16} - 1) \sim + (2^{16} - 1)$

B 2-7 采用 GCC 工具链 将下述 C 语言程序编译成汇编代码

```
void mul(long x, long y, long des) {  
    long t = sum(x,y);  
    des = t;  
}
```

- A. rsi, rcx
- B. rdi, rdx
- C. rsi, rdi
- D. rdx, rsi

D 2-8 “缺页”属于异常中的:

- A. 中断
- B. 终止
- C. 陷阱
- D. 故障

D 2-9 某些情况下 read 和 write 传送的字节数比程序要求的要少, 这些不足值出现的原因包括

- A. 读到EOF
- B. 从终端读文本行
- C. 读取网络套字节
- D. 典中典之以上都正确

D 2-10 链接可以发生在哪个阶段? 最正确的答案是

- A. 加载时
- B. 运行时
- C. 编译时
- D. 典中典之以上都正确

C 2-11 在 Linux 下, 下列哪个命令行可以将 C 程序 test.c 编译生成可执行文件

- A. objdump -d test.o
- B. gcc -o1 -o result test.o
- C. gcc -Og -o result test.c
- D. gcc -Og -c test test.c

C 2-12 表示传送字节的指令是

- A. movw
- B. movq
- C. movb
- D. movl

D 2-13 下列寄存器中，不属于被调用者保存寄存器的是

- A. r12
- B. rbp
- C. rbx
- D. rax

B 2-14 设数组 short = T[3]。假设数组的起始地址为  $x_t$ ，则 T[2] 的地址是

- A.  $x_t + 8$
- B.  $x_t + 4$
- C.  $x_t + 16$
- D.  $x_t + 2$

B 2-15 假设 p 为一个 char\* 类型的指针，他的值为 x，则表达式  $(int^*)p+7$  的值为

- A.  $4x + 28$
- B.  $4x + 7$
- C.  $x + 28$
- D.  $x + 7$

A 2-16 异常中异步发生的是

- A. 中断
- B. 故障
- C. 终止
- D. 陷阱

A 2-17 已知寄存器 rax 的值为 0x0011223344556677，执行指令 movl \$-1, %eax，则 rax 的值变为：

- A. 0x00000000FFFFFFFF
- B. 0x0000000000000000FFFF
- C. 0x00112233FFFFFFFF
- D. 0x001122330000FFFF

D 2-18 已知地址 0x100 的值为 0xff，地址 0x104 的值为 0xab，地址 0x108 的值为 0x13，地址 0x10c 的值为 0x11，寄存器 %rax 的值为 0x100，寄存器 %rdx 的值为 0x3，则操作数  $9(%rax,%rdx)$  代表的值是

- A. 0xff
- B. 0xab
- C. 0x13
- D. 0x11

C 2-19 下面关于 Unix I/O 的叙述中，不正确的是

- A. 可以读取文件的元数据
- B. 允许执行 I/O 重定向
- C. Unix I/O 可以完成的事情，标准 I/O 也可以完成
- D. 将一个打开的文件模型化为一个流

B 2-20 已知 IEEE754 单精度浮点数的值为 0xC0B00000，他的二进制是：

- A. -5.5
- B. -0.75
- C. -1.5
- D. -2.75

## 填空题

4-1 (1pts \* 10 = 10pts)

若某机器为6位，有符号数 int 用补码表示，填写下表 (1) - (10)，有如下声明：

```
int x = -1;  
unsigned ux = x;
```

表达式	十进制表示	二进制表示
零	0	000000
-	-10	(5) 110110
x	-1	(6) 111111
ux	(1) 63	(7) 111111
x>>1	(2) -1	(8) 111111
TMin	(3) -32	(9) 1000000
TMin+TMin	(4) 0	(10) 0000000

4-2(2pts\*3=6pts)

分析下表左侧汇编代码，找出右侧唯一等价的 C 函数

```
foo1:
    lea 0xf(%rdi), %eax
    test %edi, %edi
    comvns %edi, %eax
    sar $0x4, %eax
    retq
```

```
foo2:
    mov %edi, %eax
    shr $0x1f, %eax
    retq
```

```
foo3:
    mov %edi, %eax
    sal %0x4, %eax
    sub %edi, %eax
    retq
```

```
int choice1(int x) {
    return x < 0;
}
```

```
int choice2(int x){
    return (x >> 31) & 1;
}
```

```
int choice3(int x){
    return 15 * x;
}
```

```
int choice4(int x){
    return (x + 15)/4;
}
```

```
int choice5(int x){
    return x / 16;
}
```

```
int choice6(int x){
    return (x >> 31);
}
```

- 1. foo1 对应choice 5
- 2. foo2 对应choice 2
- 3. foo3 对应choice 3

4-3 求M、N (8分)

```
long mat1[M][N];
long mat2[N][M];
long sum_element(long i, long j) {
    return mat1[i][j]+mat2[i][j];
}
```

```

sum_element:
    leaq (%rdi,%rdi,2)%rdx
    addq %rsi,%rdx
    leaq (%rdi,rdi,8), %rax
    addq %rax,%rsi
    movq mat2(,%rsi,8),%rax
    addq mat1(%rdx,8),%rax
    retq

```

#### 4-4 程序完形填空 (8分)

```

foo:
    movl $0,%eax
    jmp .L2
.L3:
    subl $1,%edi
    addl $1,%esi
    addl $1,%eax
.L2:
    cmpl %esi,%edi
    jg .L3
    addl $1,%eax
    ret

```

$x - 1$   
 $y + 1$   
 $i + 1$   
 $x > y$   
~~result~~  
~~ret~~

```

int foo(int x,int y){
    int result;
    for((1);(2);result++) {
        (3);
        (4);
    }
    result++;
    return result;
}

```

$result = 0; \quad x > y;$   
 ~~$x--;$~~   
 ~~$y++;$~~

#### 4-5 阅读下列代码，回答以下问题 (12分)

调用 bar 函数时，参数 x 的值是 0xaabbccdd，且在响应 gets 时键入 '01234567890123456'，分析下表中给出的代码，然后回答以下问题。

(提示：1、get 是标准库函数，%rdi 存放的是输入缓冲区等候地址。2、x86 处理器小端存储。3、'0'-'9' 的字符 ASCII 码为 0x30-0x39)

```

push %rbp
mov %rsp,%rbp
sub $0x30,%rsp -48
mov %edi,-0x24(%rbp) -36
movl $0x11223344, -0x10(%rbp)
mov -0x24(%rbp),%eax
mov %eax, -0xc(%rbp)
lea -0x20(%rbp),%rax
mov %rax,%rdi
mov $0x0,%eax
callq 0x400450<gets@plt>
mov -0xc(%rbp),%edx
mov -0x10(%rbp),%eax
lea -0x20(%rbp),%rcx
mov %eax,%esi
mov $0x400654,%edi
mov $0x0,%eax
callq 0x400430 <printf@plt>
nop
mov %rbp,%rsp
pop %rbp
retq

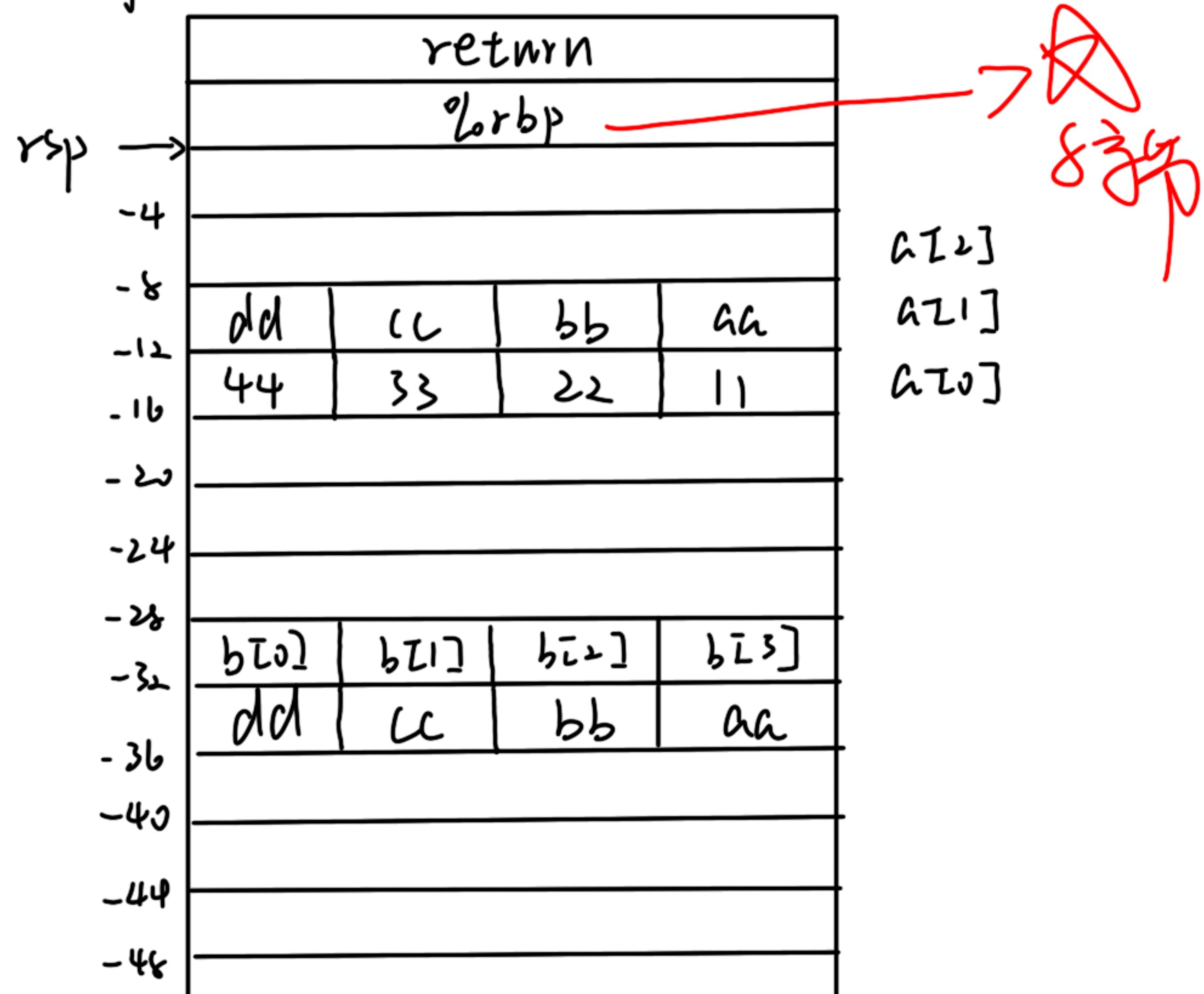
```

$rdi: -32$

```

void bar(int x) {
    int a[3];
    char buf[4];
    a[0] = 0x11223344;
    a[1] = x;
    gets(buf);
    printf("a[0]=0x%x,a[1]=0x%x,buf=%s\n",a[0],a[1],buf);
}

```



- a) 指出程序中下列值是否被溢出破坏(2pts\*4=8pts)

- a[0] (y/n) ✓
- a[1] (y/n) ✗
- x 的值 (y/n) ✗
- 寄存器 %rbp 保存的值 (y/n) ✗

b) 程序打印结果为(1pts\*3=3pts)

- a[0]:0x 1122003b
- a[1]:0x aabbccdd
- buf(ASCII): 01234567890123456

c) 如果编写C程序调用上述bar函数，问： (1pts)

- 当输入到第39个字符后无法从bar函数正常返回。

40