

CSAPP 某年期末

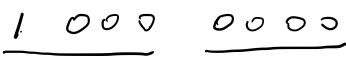
判断题 2pts * 8=16pts

- 1-1 (x|-x)>>31 == -1 x 的类型是 int ☒
- 1-2 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。 ☒
- 1-3 -1 < 0U ☒
- 1-4 cmpq (%rax), 6(%rsp) ☒
- 1-5 fork 调用一次，返回两次 ☒
- 1-6 x>>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) ☒ lea 目的寄存器

单选题 2pts * 20 = 40pts

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

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

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

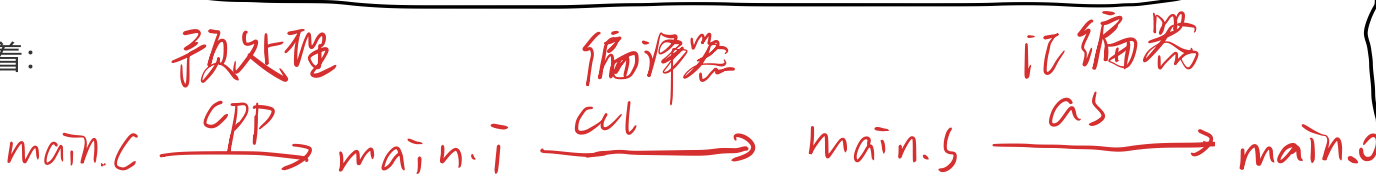
-128~127
0~255

$x+y=30 = 16+14 = 1E$
OF

OF 无符号
OF 有符号

2-4 对于 gcc 工具链而言, cpp、ccl、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

链接器 ld → main. shell 调用 loader (加载器) → 执行.

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

400 403
| 67 45 23 01 |

2-6 十六位补码整数的范围

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

16位.

$-2^{15} \sim 2^{15}-1$

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

2-8 “缺页”属于异常中的：

- A. 中断
- B. 终止
- C. 陷阱
- D. 故障

设备 I/O. 异步
致命错误. 同步. 原进程停止
syscall. 内核. 同步. 返回原进程
缺页. 用户进程. 同步

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

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

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

gcc -o 可执行文件

gcc -S: 汇编文件

gcc -c: 可重定位文件

objdump -d: 反汇编

2-12 表示传送字节的指令是

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

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

rbx rbp r12~r15: 被调用者保存

其他除rsp外: 调用者保存

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. $x_t + 2$

$x_t + 8$

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. $x + 7$

$(int*)p + 7$ +28

$x + 28$

2-16 异常中异步发生的是

- A. 中断
- B. 故障
- C. 终止
- D. 陷阱

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. `0x00000000000000FFFF`
- C. `0x00112233FFFFFFFF`
- D. `0x00112233000000FFFF`

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`

$rax + rdx + 9 = 103 + 9 = 10C$

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

元数据 × 网络套字串 ×

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

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

$C=12 \quad B=11$

$E = exp - bias$

$12 = 1100 \quad 11 = 1011$

$\frac{1}{100} \frac{000}{000} \frac{1011}{1011} \frac{0000}{0000}$

$exp = 256 + 1$

-256

$E = 128 + 1 - 127 = 2$

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：

$-1.011 \times 2^{34} = -101.1$

$1 + \frac{1}{4} + \frac{1}{2} = 1.5$

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


美

```
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
```

$$\begin{aligned} rdx &= 3i + j \\ rax &= 9i \\ j &+= 9i \\ rax &= \underline{mat2 + 8j} + \underline{mat1 + 8(3i + j)} \end{aligned}$$

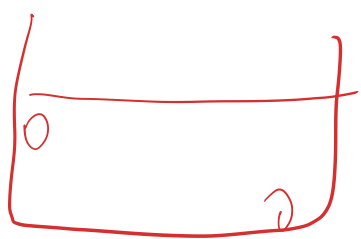
4-4 程序完形填空 (8分)

```
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
```

$eax = 0$
 $L2:$
 $x--$
 $y++$
 $eax++$
 $if(x > y) .L3$
 $else\ eax++$

$$\begin{aligned} &mat2 + 8(9i + j) \qquad \underline{mat1 + 24i + 8j} \\ &= \underline{mat2 + 72i + 8j} \qquad \textcircled{3} \end{aligned}$$

```
int foo(int x,int y){
    int result;
    for((1)  $y=0$ ; (2)  $x=y$ ; result++) {
        (3)  $x--$ ;
        (4)  $y++$ ;
    }
    result++;
    return result;
}
```

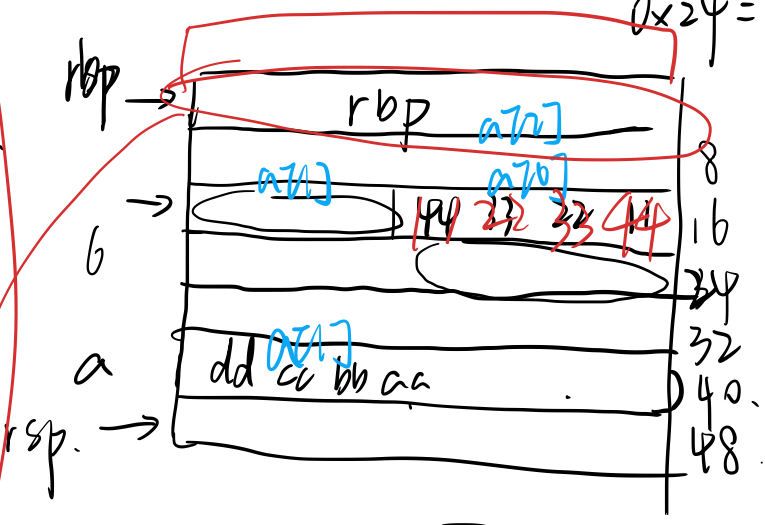


4-5 阅读下列代码，回答以下问题 (12分)

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
mov %edi,-0x24(%rbp)
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
```

$0x30 = 16 \times 3 = 6 \times 8$
 $rbp = rsp$



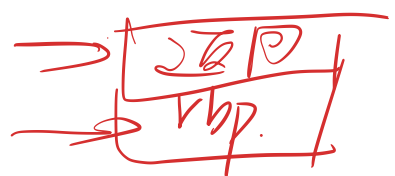
$$0x24 = 4 + 32 = 36$$

$0x10 = 16$
 $0x24 = 36$
 $eax = x$
 12

$eax = 0$
 $0 =$

$0x30 = 48$
 $0x24 = 36$
 $0x10 = 16$
 $C = 12$
 $0x20 = 32$

$rax = ??$
 $x = \text{rdi}$
 $eax = 0$

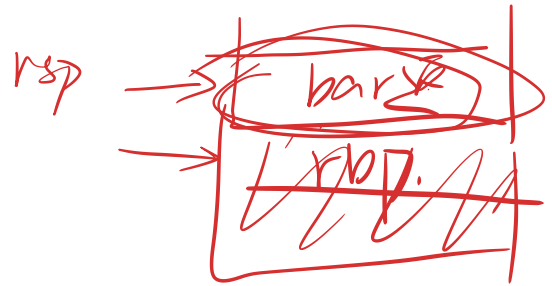


```
void bar(int x) {
    int a[3];
    char buf[4];
    a[0] = 0x11223344;
    a[1] = x;
    gets(buf);
    printf("a[0]=0x%x,a[1]=0x%x,buf=%s\n",a[0],a[1],buf);
}
```

rbp
返回

main()
bar()

a) 指出程序中下列值是否被溢出破坏(2pts*4=8pts)



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

b) 程序打印结果为(1pts*3=3pts)

- a[0]:0x 38 39 30 31
- a[1]:0x aa bb cc dd
- buf(ASCII): 30 31 32 }

c) 如果编写C程序调用上述bar函数，问： （1pts）

- 当输入到第 25 个字符后无法从bar函数正常返回。