华东师范大学期中试卷（A）答案

2015 —2016 学年第 二学期

课程名称：__计算机系统____

学生姓名：_____ 学 号：_____

专 业：_____ 年级/班级：_____

课程性质：公共必修、公共选修、**专业必修**、专业选修

一	二	三	四	五	六	七	八	总分	**阅卷人签名**

一、选择题（每小题 2 分，共 30 分）

1、我们通常所说的“字节”由___B__个二进制位构成。

A、1 B、8 C、16 D、32

2、IA32 的 CPU 中有一个程序计数器(又称指令计数器)。它用于存储_____A_____。

A. 保存将要执行的下一条指令在主存中的地址

B. 保存当前 CPU 所要访问的内存单元地址

C. 暂时存放 ALU 运算结果的信息

D. 保存当前正在执行的一条指令

3、逻辑运算 ！！0x21 的结果用十六进制表示为__D___ 。

A. 0X00 B. 0X21 C. 0X12 D. 0X01

4、考虑程序片段：short int x=-12345; unsigned short int y=x; 则 y 的真值为_____D_____。

A. -12345 B. 12345 C. 53190 D. 53191

5、位移运算：对参数 x =0x85,则 x>>4(算术右移)的结果是___C___。

A. 0x08 B. 0x50 C. 0xf8 D. 0x5f

6、sizeof（int）的值为_____B_____。

A. 2 B. 4 C. 8 D. 4 或 8

7、定义变量：int x=0x01234567；采用小端法存储数据，x 的起始地址为 0x1000，则 x 的存储形式为_____A_____。

A.

地址：	0x1000	0x1001	0x1002	0x1003	
	0x67	0x45	0x23	0x01	

B.

地址：	0x1000	0x1001	0x1002	0x1003	
	0x01	0x23	0x45	0x67	

C.

地址：	0x1000	0x1001	0x1002	0x1003	
	0x45	0x67	0x01	0x23	

D.

地址：	0x1000	0x1001	0x1002	0x1003	
	0x67	0x45	0x00	0x00	

8、（3.14+1e10）-1e10 在计算机中的运算结果为_____D_____。

A. 0　　B. 3.14　　C. 1e10　　D. 0.0

9、R[Ea]寻址方式属于__B___ 。

A.立即数寻址　　B.寄存器寻址　　C.绝对寻址　　D.间接寻址

10、假设初始值：%dh=CD，%eax=98765432 则执行指令（MOVB %dh，%al）后，%eax的值为多少？____A____

A. %eax= 987654CD　　B. %eax= CD765432

C. %eax= FFFFFFCD　　D. %eax= 000000CD

11、假设寄存器%eax 的值为 x，%ecx 的值为 y，则执行汇编指令 Leal （%eax ，%ecx，2），%edx 后，存储在寄存器%edx 中的值为____C____。

A. 2x　　B. 2y　　C. x+2y　　D.2x+y

12、条件码 ZF 表示__A____ 。

A. 零标志　　B.符号标志　　C.溢出标志　　D.进位标志

13、某硬盘有五个盘片，每个扇区 512 字节，每个面 20000 条磁道，每条磁道平均 400 个扇区，则这个磁盘容量是____B_____（GB）。

A. 20.48　　B. 40.96　　C. 163.84　　D. 327.68

14、假设某磁盘的旋转速率为 15000RPM，$T_{avg\ seek}$=8ms，每条磁道的平均扇区数为 500，则这个磁盘上一个扇区的访问时间为____C_____ms。

A. 8　　B. 9　　C. 10　　D. 11

15、与传统旋转式硬盘相比，下面不是固态硬盘的特点的是_____C_____。

A. 更快速　　B.低能耗　　C.读写速度相当　　D.写访问比读访问慢 10 倍

二、填空题（每小题 2 分，共 10 分）

1、在 IA32 体系结构机器中，存储 Long int 型变量需要__4__个字节，而 X86-64 体系结构机器中，存储 Long int 型变量又需要__8__个字节。

2、处理同样字长的有符号数和无符号数之间的转换规则是：位模式__不变__（改变/不变），数值重新解析。

3、浮点加法运算__满足__（满足/不满足）交换律，__不满足__（满足/不满足）结合律。

4、IA32 架构机器中，执行 push %eax 指令，栈指针（%esp）__减__（加/减）4 后，再存储%eax 的值到%esp 位置。

5、嵌套数组中元素采用____行优先____（行优先/列优先）顺序排列。

三、分析、计算题（共 60 分）

1、（8 分）假设某存储器及寄存器存储的数据如下表所示：

存储器		寄存器	
地址	值	寄存器名称	值
0x200	0xFF	%eax	0x200
0x204	0xAB	%ecx	0x1
0x208	0x13	%edx	0x3
0x20c	0x11		

试给出如下所示操作数的值。

操作数	值
%eax	（1）0X200
(%eax)	（2）0XFF
0x204	（3）0XAB
$0x204	（4）0X204
0x1fc(, %ecx,4)	（5）0XFF
(%eax,%edx,4)	（6）0X11
1(%eax,%edx)	（7）0XAB
516(%ecx,%edx)	（8）0X13

2、（8 分）考虑一个 8 位补码表示的有符号整型数据，填写下表（基于 8 位补码加减法运算规则执行加减法运算）：

数	二进制表示
Zero（零）	（1）0000 0000
TMax（最大有符号整数）	（2）0111 1111
TMin（最小有符号整数）	（3）1000 0000
TMin + TMin	（4）0000 0000
TMin+1	（5）1000 0001
TMax+1	（6）1000 0000
-TMax	（7）1000 0001
-TMin	（8）1000 0000

3、（8 分）定义变量：float x；试填写下表：

x	二进制表示	十进制表示
最大规格化正数	（1）0 11111110 111…111	（2）$(2-2^{-23})*2^{127}$
最小规格化正数	（3）0 00000001 000…000	（4）$1*2^{-126}$
最大规格化负数	（5）1 00000001 000…000	（6）$-1*2^{-126}$
最小规格化负数	（7）1 11111110 111…111	（8）$-(2-2^{-23})*2^{127}$

4、（15 分）库函数中 memcpy 和 copy_from_kernal 的定义如下所示：

```
/* Declaration of library function memcpy */
void *memcpy(void *dest, void *src, unsigned int    n);
/* Kernel memory region holding user-accessible data */
#define KSIZE 1024
char kbuf[KSIZE];
/* Copy at most maxlen bytes from kernel region to user buffer */
int copy_from_kernel(void *user_dest, int maxlen) {
    /* Byte count len is minimum of buffer size and maxlen */
    int len = KSIZE < maxlen ? KSIZE : maxlen;
    memcpy(user_dest, kbuf, len);
    return len;
}
```

某应用程序片段如下所示：

```
#define MSIZE 528
void getstuff( ) {
    char mybuf[MSIZE];
    copy_from_kernel(mybuf,   -MSIZE);
    . . .
}
```

（1） 试分析该程序的潜在危害

（2） 如何消除其潜在危害？

答案：（1）函数 copy_from_kernel 是要将一些操作系统内核维护的数据（最多 1024 字节）复制到指定的用户可以访问的存储器区域。假设有些怀有恶毒的程序员在调用 copy_from_kernel 的代码中对 maxlen 使用了负数值，则会把这个值赋给 len ，然后 len 会作为参数 n 被传递给 memcpy（参数 n 是被声明为数据类型 unsigned int 的），则 memcpy 会把它当作一个非常大的正整数，并且试图将这样多字节的数据从内核区域复制到用户的缓冲区，这样程序还是能读到没有被授权的内核存储器区域，发生信息泄露。

（2）将 copy_from_kernel 的参数 maxlen 声明为类型 unsigned int ，也就是与 memcpy 的参数 n 一致。同时，将本地变量 len 和返回值声明为 unsigned int。

5、 (12 分) 考虑如下 C 程序片段，其中去掉了 switch 语句的主体，请根据给出的汇编代码及跳转表，补充完整 C 语言程序片段中的 switch 语句主体。

```
long switch_eg(long x, long y, long z)
{
    long w = 1;
    switch(x) {
      /*      Fill in code here */
      . . .
    }
    return w;
}
```

汇编代码如下：

```
switch_eg:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax      # eax = x
    cmpl    $6, %eax
    ja      .L2
    jmp     *.L7(,%eax,4)
.L2:
    movl    $2, %eax
    jmp     .L8
.L5:
    movl    $1, %eax
    jmp     .L9
.L3:
    movl    16(%ebp), %eax     # eax = z
    imull   12(%ebp), %eax
```

```
    jmp     .L8
.L4:
    movl    12(%ebp), %edx    # edx = y
    movl    %edx, %eax
    sarl    $31, %edx
    idivl   16(%ebp)          # R[%edx]:R[%eax] mod S
.L9:
    addl    16(%ebp), %eax
    jmp     .L8
.L6:
    movl    $1, %eax
    subl    16(%ebp), %eax
.L8:
    popl    %ebp
    ret
```

跳转表如下：

```
.section   .rodata
        .align 4
.L7:
    .long   .L2
    .long   .L3
    .long   .L4
    .long   .L5
    .long   .L2
    .long   .L6
    .long   .L6
```

答案：

```
long switch_eg(long x, long y, long z)
{
    long w = 1;
    switch(x) {
    case 1:
        w = y*z;
        break;
    case 2:
        w = y/z;
        /* Fall Through */
    case 3:
        w += z;
        break;
    case 5:
    case 6:
        w - = z;
```

```
            break;
        default:
            w = 2;
          }
        return w;
}
```

6、（9 分）考虑如下 C 程序片段：

```
int main(void) {
   union {
          unsigned char c[8];
          unsigned short s[4];
          unsigned int i[2];
          unsigned long k[1];
       } dw;
int j;
for (j = 0; j < 8; j++)
       dw.c[j] = 0xc0 + j;
printf("Shorts 0-3=[0x%x,0x%x,0x%x,0x%x]\n", dw.s[0], dw.s[1], dw.s[2], dw.s[3]);
printf("Ints 0-1= [0x%x,0x%x]\n", dw.i[0], dw.i[1]);
printf("Long 0= [0x%lx]\n", dw.k[0]);
}
```

试分析该程序在 IA32 架构机器中运行后的输出结果。

答案：

Shorts 0-3 == [0xc1c0,0xc3c2,0xc5c4,0xc7c6]

Ints 0-1 == [0xc3c2c1c0,0xc7c6c5c4]

Long 0 == [0x c3c2c1c0]