Problem 1. （40 points）

1. Consider the following code, what is the output of the printf?

```
int x = 0xF1F2F3F4 >> 16;
char y = (char) x;
unsigned char z = (unsigned char) x;
printf("%d, %u", y, z);
```

(a) -12, 244 (b) -13, 243 (c) -14, 242 (d) -15, 241

2. In two's compliment, what is `TMax+1`

(a) TMax (b) TMin (c) 0 (d) −1

For question 3/4/5 suppose that x and y have byte values 0x66 and 0x39

3. what is `x|y`

(a) 0x00 (b) 0x01 (c) 0x46 (d) 0x7F

4. what is `x&&y`

(a) 0x00 (b) 0x01 (c) 0x46 (d) 0x7F

5. what is `x&!y`

(a) 0x00 (b) 0x01 (c) 0x46 (d) 0x7F

6. Select the two's complement negation (补码负值) of the following binary value: 0100

(a) 11000100 (b) 10111011 (c) 10111100 (d) 10111010

7. In C, the expression `-1 < 0U` evaluates to:

(a) True (1) (b) False (0)

8. what is `uy`

```
short sx = -12345;  // 0xcfc7
unsigned uy = sx;
```

(a) `0xffffcfc7` (b) `0x0000cfc7`

9. Let int `x = -31/4` and int `y = -31>>2`. What are the values of x and y?

(a) x = −7, y = −7 (b) x = −8, y = −8 (c) x = −8, y = −7 (d) x = −7, y = −8

10. Assume variables x, f, and d are of type int, float, and double, respectively. Their values are arbitrary, except that neither f nor d equals +∞, −∞, or NaN. Which C expression will always be true?

(a) `x == (int)(float)x` (b) `d == (double)(float)d`

(c) `d*d >= 0.0` (d) `(f+d)-f == d`

11. What is the difference between the %rbx and the %ebx register on an x86 64 machine?

(a) nothing, they are the same register

(b) %ebx refers to only the low order 32 bits of the %rbx register

(c) they are totally different registers

(d) %ebx refers to only the high order 32 bits of the %rbx register

12. On IA32 systems, where is the value of old %ebp saved in relation to the current value of %ebp

(a) there is no relation between them

(b) old %ebp is stored at (%ebp - 4)

(c) old %ebp is stored at (%ebp + 4)

(d) old %ebp is stored at (%ebp)

13. If a `direct mapped` cache is 8KB in size, and has 32 byte cache blocks, how many lines are there in `each` set?

(a) 256 (b) 64 (c) 32 (d) 1

14. You have a 32-bit virtual memory system with 4KB page frames, with a TLB with 4 set each of which is 8-way set associative. How many bits of the virtual address form the TLBi (TLB index)?

(a) 2 (b) 4 (c) 8 (d) 12

15. Which section of an ELF file contains the compiled functions from a program?

(a) .data (b) .rodata (c) .text (d) .bss

16. When it succeeds, `execve` is called once and returns how many times?

(a) 0 (b) 1 (c) 2 (d) 1 or multiple times

17. How many "hello" output lines does this program print?

```
void doit()
{
      if (Fork() == 0) {
          Fork();
          printf("hello\n");
          exit(0);
      }
      return;
}
```

```
int main()
{
      doit();
      printf("hello\n");
      exit(0);
}
```

(a) 6 (b) 5 (c) 4 (d) 3

18. How many "hello" output lines does this program print?

```
void doit()
{
      if (Fork() == 0) {
          Fork();
          printf("hello\n");
          return;
      }
      return;
}
```

```
int main()
{
      doit();
      printf("hello\n");
      exit(0);
}
```

(a) 6 (b) 5 (c) 4 (d) 3

19. Linux virtual memory area can be mapped to

(a) regular file in the Unix file system only (b) anonymous file only

(c) either regular file or anonymous file (d) nothing

20. `task_struct` for each process in Linux is stored in

(a) code segment (b) data segment

(c) shared library segment (d) kernel virtual memory

Problem 2. （60 points）

1. Consider the following 5-bit floating point representation based on the IEEE floating point format. This format does not have a sign bit – it can only represent nonnegative numbers.

- There are k = 3 exponent bits. The exponent bias is 3. 有 k = 3 个阶码位，偏置值是 3
- There are n = 2 fraction bits. 有 n = 2 个小数位

Below, you are given some decimal values, and your task is to encode them in floating point format. In addition, you should give the rounded value of the encoded floating point number. To get credit, you must give these as whole numbers (e.g., 17) or as fractions in reduced form (e.g., 3/4). Any rounding of the significand is based on round-to-even. 将十进制数值编码为浮点格式，舍入到偶数并给出舍入后的值，用整数（例如 17）或者分数（例如 3/4）表示。

Value	Floating Point Bits	Rounded value
1	011 00	1
7/8		
15/16		
1/16		
9/32		
15		

2. In the following code, A and B are constants defined with #define:

```
typedef struct {
    short x[A][B]; /* Unknown constants A and B */
    int y;
} str1;

typedef struct {
    char array[B];
    int t;
    short s[B];
    int u;
} str2;

void setVal(str1 *p, str2 *q) {
    int v1 = q->t;
    int v2 = q->u;
    p->y = v1+v2;
}
```

gcc generates the following code for the body of setVal:

```
movl  12(%ebp), %eax
movl  28(%eax), %edx
addl  8(%eax), %edx
movl  8(%ebp), %eax
movl  %edx, 44(%eax)
```

(1) What's the value of A?

(a) 3 (b) 5 (c) 6 (d) 7

(2) What's the value of B?

(a) 3 (b) 5 (c) 6 (d) 7

3. Which function computes the parity of argument x? 哪个函数计算参数 x 的奇偶性
(a) fun_a (b) fun_b

```
int fun_a(unsigned x) {
    int val = 0;
    while ( ________ ) {
        ________;
    }
    return ________;
}
```

assembly code:

```
x at %ebp+8
    movl    8(%ebp), %edx
    movl    $0, %eax
    testl   %edx, %edx
    je      .L7
  .L10:
    xorl    %edx, %eax
    shrl    %edx
    jne     .L10
  .L7:
    andl    $1, %eax
```

```
int fun_b(unsigned x) {
    int val = 0;
    int i;
    for ( ________ ; ________ ; ________ ) {
        ________
    }
    return val;
}
```

assembly code:

```
x at %ebp+8
    movl    8(%ebp), %ebx
    movl    $0, %eax
    movl    $0, %ecx
  .L13:
    leal    (%eax,%eax), %edx
    movl    %ebx, %eax
    andl    $1, %eax
    orl     %edx, %eax
    shrl    %ebx
    addl    $1, %ecx
    cmpl    $32, %ecx
    jne     .L13
```

4. Which value will be corrupted if user enter string “123456789012” when echo is called?

(a) saved value of %ebp

(b) saved value of %ebx (c) both (d) none

```
/* Read input line and write it back */
void echo()
{
    char buf[8];  /* Way too small! */
    gets(buf);
    puts(buf);
}
```

```
echo:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %ebx
    subl    $20, %esp
    leal    -12(%ebp), %ebx
    movl    %ebx, (%esp)
    call    gets
    movl    %ebx, (%esp)
    call    puts
    addl    $20, %esp
    popl    %ebx
    popl    %ebp
    ret
```

5. Consider the following function for computing the product of an array of n floats. We have unrolled the loop by a factor of 3.

```
float aprod(float a[], int n)
{
    int i;
    float x, y, z;
    float r = 1;
    for (i = 0; i < n-2; i+= 3) {
         x = a[i]; y = a[i+1]; z = a[i+2];
         r = r * x * y * z;  // Product computation
    }
    for (; i < n; i++)
         r *= a[i];
    return r;
}
```

For the line labeled Product computation, we can use parentheses to create different associations of the computation, as follows:

```
r = ((r * x) * y) * z;  // A1
r = r * ((x * y) * z);  // A2
r = (r * x) * (y * z);  // A3
```

Which one has the lowest CPE?

(a) A1 (b) A2 (c) A3

6. Consider the following transpose routine

```
typedef int array[2][2];
void transpose(array dst, array src) {
    int i, j;
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 2; j++) {
            dst[j][i] = src[i][j];
        }
    }
}
```

running on a machine with the following properties:

- `sizeof(int) == 4`.
- The `src` array starts at address 0 and the `dst` array starts at address 16 (decimal).
- There is a single L1 cache that is direct mapped and write-allocate, with a block size of 8 bytes.

- Accesses to the `src` and `dst` arrays are the only sources of read and write misses, respectively.
- The cache is initially empty.

(1) Suppose the cache has a total size of 16 data bytes

(2) Suppose the cache has a total size of 32 data bytes

What's the `hit` rate of `src` array and `dst` array?

(a) `0` (b) `1/4` (c) `1/2` (d) 1

7. This problem deals with virtual memory address translation using a multi-level page table, in particular the 2-level page table for a 32-bit Intel system with 4 KByte pages tables.

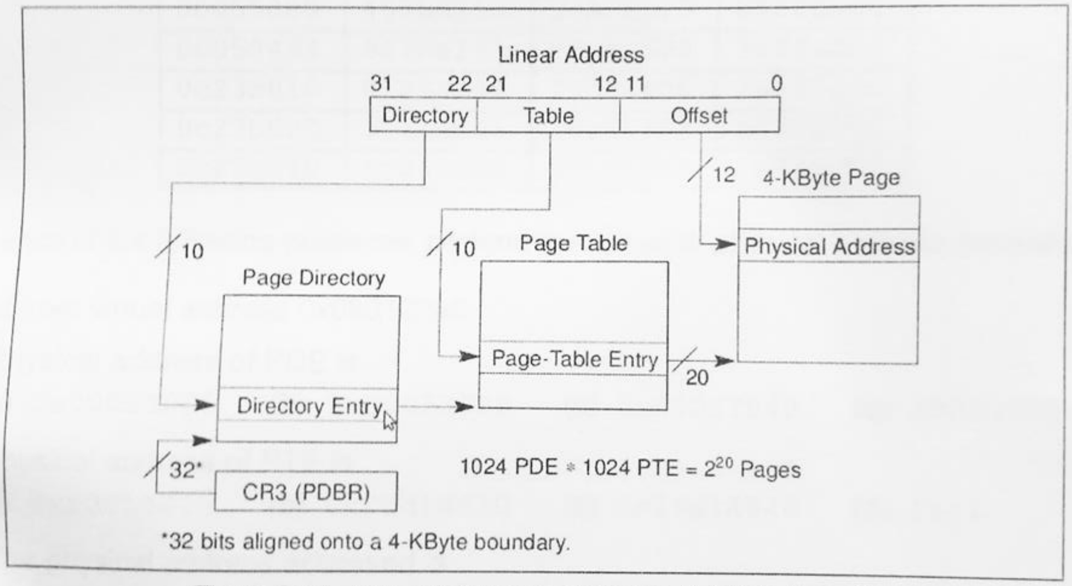

Figure 3-12. Linear Address Translation (4-KByte Pages)

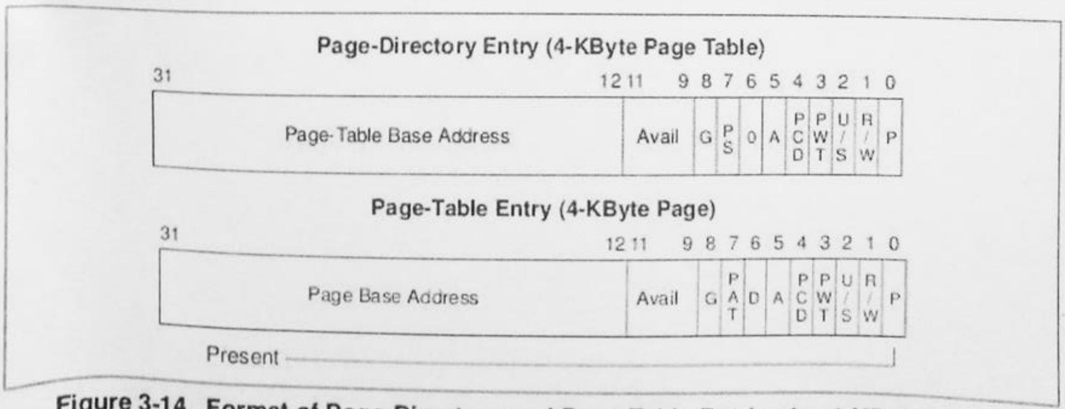

Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

The contents of the relevant sections of memory are shown on this table. All numbers are given in hexadecimal. Any memory not shown can be assumed to be zero. The Page Directory Base Address is 0x00023000.

Address	Contents	Address	Contents
00023000	beefbee0	0c23b080	0ff2d303
00023020	00055d21	0c23b274	00023d03
00023040	0c23b933	0c23b7bc	514d2274
00023080	25d14883	25d14010	0fdc1223
00023FFF	bcdeff29	25d14040	d21345a9
00055010	8974d003	25d144a0	d388bcbd
00055040	457bc293	25d14890	00b32d00
00055080	457bd293	24AEE520	b58cdad1
00055464	457be293	29DE2504	56ffad02
0c23b010	01288b52	29DE4400	2ab45cd0
0c23b020	012aab53	29DE9402	d4732000
0c23b040	00055d01	29DEE500	1a23cdb0

For each of the following problems, perform the virtual to physical address translation.

Read from virtual address 0x080102a5:

(1) Physical address of PDE is

(a) 0x00023000 (b) 0x00023020 (c) 0x00023040 (d) 00023080

(2) Physical address of PTE is

(a) 0x00055010 (b) 0x25d14010 (c) 0x25d14040 (d) fail

(3) The physical address accessed is

(a) 0x0fdc12a5 (b) 0xd21342a5 (c) 0x8974d2a5 (d) fail