北京航空航天大学

2015－2016 学年 第一学期期末

《计算机组成与体系结构》

考 试 试 卷

班 级________________学 号 __________

姓 名________________成 绩 __________

2016 年 1 月 12 日

北京航空航天大学 2015－2016 学年 第一学期 《计算机组成与体系结构》期末考试

学号__________ 姓名 ___________

答题页 1

注意：答案全部填写在答题页中，其它地方无效

Problem 1.（20 points）

1_____ 2_____ 3_____ 4_____ 5_____

6_____ 7_____ 8_____ 9_____ 10_____

Problem 2.（12 points）

Format A		Format B	
Bits	Value	Bits	Value
010 1110			
110 1111			
000 0001			
			16

Problem 3.（10 points）

A: x = _______

B: string "0123456789" is stored at ____________

C: buf[0] = 0x_____ _____ _____ _____

buf[1] = 0x_____ _____ _____ _____

buf[4] = 0x_____ _____ _____ _____

buf[5] = 0x_____ _____ _____ _____

D: Value at %esp is ____________

E: ____

Problem 4.（10 points）

M = _________ N = _________

学号__________ 姓名 ___________

答题页 2

Problem 5.（10 points）

A1_____ A2_____ A3_____ A4_____ A5_____

Problem 6.（6 points）

A

dst array		
	col 0	col 1
row 0	m	
row 1		

src array		
	col 0	col 1
row 0	m	
row 1		

B

dst array		
	col 0	col 1
row 0	m	
row 1		

src array		
	col 0	col 1
row 0	m	
row 1		

Problem 7.（10 points）

a=______, b=______, c=______, a=______, c=______

Problem 8.（10 points）（fill in Y or N）

A_____ B_____ C_____ D_____ E_____

Problem 9.（12 points）

1. Read from virtual address `0x7bcd8001`:
   a. Physical address of PDE: ____________
   b. Physical address of PTE: ____________
   c. The physical address accessed is ____________
2. Read from virtual address `0x04002abc`:
   a. Physical address of PDE: ____________
   b. Physical address of PTE: ____________
   c. The physical address accessed is ____________

Problem 1. （20 points）

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

```
int x = 0x152F2F10 >> 12;
char y = (char) x;
unsigned char z = (unsigned char) x;
printf("%d, %u", y, z);
```

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

2. In two’s compliment, what is T_{Max} +1?

(a) T_{Min} (b) T_{Max}
(c) 0 (d) −1

3. Let `int x = −17/4` and `int y = −17 >> 2`. What are the values of x and y?

(a) x = −5, y = −5
(b) x = −4, y = −4
(c) x = −5, y = −4
(d) x = −4, y = −5

向0
向下

4. In C, which one is true?

(a) `10000U > -1`
(b) `for double d,  d * d >= 0`
(c) `for int x,  x * x >= 0`
(d) `for int x,  x == (int) (float) x`

5. By default, on Intel x86, the stack

(a) Is located at the bottom of memory
(b) Grows down towards smaller addresses
(c) Grows up towards larger addresses
(d) Is located in the heap

6. Intel x86-64 systems are

(a) Little endian
(b) Big endian
(c) Have no endianess
(d) Depend on the operating system

7. Select the two's complement negation of the following binary value: 00001011:

(a) 11110100 (b) 11110111

(c) 11110101 (d) 10001011

取反+1

8. Which of the following is not a default action for any signal type?

(a) The process terminates

(b) The process reaps the zombies in the waitlist

(c) The process stops until restarted by a SIGCONT signal

(d) The process ignores the signal

(e) The process terminates and dumps core

9. When it suceeds, **`fork`** is called once and returns how many times?

(a) 0 (b) 1

(c) 2 (d) 3

10. A system uses a four-way set-associative cache with 16 sets and 32-byte blocks. Which set does the byte with the address 0xdeadbeef map to?

(a) Set 7

(b) Set 11

(c) Set 13

(d) Set 14

Problem 2. （12 points）

Consider the following two 7-bit floating-point representations based on the IEEE floating point format. Neither has a sign bit - they can only represent nonnegative numbers.（无符号位的7位IEEE浮点格式）

1.Format A

- There are `k=3` exponent bits. The exponent bias is 3.
- There are `n=4` fraction bits.

2.Format B

- There are `k=4` exponent bits. The exponent bias is 7.
- There are `n=3` fraction bits.

Fill in the blanks to convert given value in one format to the closest value in another format. If necessary, you should apply the round-to-even rounding rule. In addition, give the values of numbers given by the Format A and Format B bit patterns. Give these as whole numbers (e.g., 17) or as fractions (e.g., 17/64)

填空，将给出的一种格式的值转换为另一种格式，如果需要，使用舍入到偶数的原则。位模式对应的数值用整数（如17）或者小数（如17/64）表示。

Format A		Format B	
Bits	Value	Bits	Value
010 1110			
110 1111			
000 0001			
			16

Problem 3. （10 points）

This problem concerns the following C code, compiled on a 32-bit machine:

```
void foo(char * str, int a) {
   int buf[2];
   a = a;     /* Keep GCC happy */
   strcpy((char *) buf, str);
}

/* The base pointer for the stack frame of caller() is:
0xffffd3a0
*/
void caller() {
     foo(``0123456789'', 0xdeadbeef);
}
```

Here is the corresponding machine code on a 32-bit Linux/x86 machine:

```
080483c8 <foo>:
080483c8 <foo+0>:      push  %ebp
080483c9 <foo+1>:      mov   %esp,%ebp
080483cb <foo+3>:      sub   $0x18,%esp
080483ce <foo+6>:      lea   -0x10(%ebp),%edx
080483d1 <foo+9>:      mov   0x8(%ebp),%eax
080483d4 <foo+12>:     mov   %eax,0x4(%esp)
080483d8 <foo+16>:     mov   %edx,(%esp)
080483db <foo+19>:     call  0x80482c0 <strcpy@plt>
080483e0 <foo+24>:     leave
080483e1 <foo+25>:     ret

080483e2 <caller>:
080483e2 <caller+0>:   push  %ebp
080483e3 <caller+1>:   mov   %esp,%ebp
080483e5 <caller+3>:   sub   $0x8,%esp
080483e8 <caller+6>:   movl  $0xdeadbeef,0x4(%esp)
080483f0 <caller+14>:  movl  $0x80485d0,(%esp)
080483f7 <caller+21>:  call  0x80483c8 <foo>
080483fc <caller+26>:  leave
080483fd <caller+27>:  ret
```

Here are some notes to help you work the problem:

• `strcpy(char *dst, char *src)` copies the string at address `src` (including the terminating `'\0'` character) to address `dst`.

• Keep endianness in mind.

• You will need to know the hex values of the following characters:

Charac	Hex	Charac	Hex
'0'	0x30	'5'	0x35
'1'	0x31	'6'	0x36
'2'	0x32	'7'	0x37
'3'	0x33	'8'	0x38
'4'	0x34	'9'	0x39

Now consider what happens on a Linux/x86 machine when `caller` calls `foo`.

A. Just before `foo` calls `strcpy`, what integer x, if any, can you guarantee that buf[x] == a ?
foo 调用 strcpy 之前，使得 buf[x] == a 的整数 x 值为多少？

B. At what memory address is the string “0123456789” stored (before it is strcpy’d)?
在被 strcpy 拷贝之前，字符串“0123456789”保存在哪个地址处？

C. Just after `strcpy` returns to `foo`, fill in the following with hex values:
strcpy 返回到 foo 之后，在答题页中填写各地址处的十六进制值。

D. Immediately before `foo`’s ret call, what is the value at %esp (what’s on the top of the stack)?
foo 中 ret 指令调用前，%esp 处的值是（栈顶是什么）？

E. Will a function that calls `caller()` segfault or notice any stack corruption? Explain.
调用 caller()的函数会段错误或栈损坏吗？请解释。

Problem 4. （10 points）

Consider the source code below, where `M` and `N` are constants declared with `#define`.

```
int array1[M][N];
int array2[N][M];
int copy(int i, int j)
{
   array1[i][j] = array2[j][i];
}
```

Suppose the above code generates the following assembly code:

```
copy:
  pushl  %ebp
  movl   %esp, %ebp
  pushl  %ebx
  movl   8(%ebp), %ecx
  movl   12(%ebp), %ebx
  leal   (%ecx,%ecx,8), %edx
  sall   $2, %edx
  movl   %ebx, %eax
  sall   $4, %eax
  subl   %ebx, %eax
  sall   $2, %eax
  movl   array2(%eax, %ecx, 4), %eax
  movl   %eax,array1(%edx, %ebx, 4)
  popl   %ebx
  movl   %ebp, %esp
  popl   %ebp
  ret
```

What are the values of `M` and `N`?

Problem 5. （10 points）

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

```
int aprod(int a[], int n)
{
   int i, x, y, z;
   int 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 5 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
r = r * (x * (y * z));   // A4
r = (r * x) * (y * z);   // A5
```

Determine the lower bound on the CPE set by the data dependencies.

Assume we run these functions on a machine where latency is shown in the following table.

	Integer		Single-precision		Double-precision	
Operation	Latency	Issue	Latency	Issue	Latency	Issue
Addition	1	0.33	3	1	3	1
Multiplication	3	1	4	1	5	1
Division	11–21	5–13	10–15	6–11	10–23	6–19

The lower bound of CPE:

A1_____ A2_____ A3_____ A4_____ A5_____

Problem 6. （6 points）

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[i][j] = src[j][i];
        }
    }
}
```

running on a hypothetical 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.

A. Suppose the cache has a total size of 16 data bytes (i.e., the block size times the number of sets is 16 bytes) and that the cache is initially empty. Then for each row and col, indicate whether each access to `src[row][col]` and `dst[row][col]` is a hit (h) or a miss (m). For example, reading `src[0][0]` is a miss and writing `dst[0][0]` is also a miss.

dst array		
	col 0	col 1
row 0	m	
row 1		

src array		
	col 0	col 1
row 0	m	
row 1		

B. Repeat part A for a cache with a total size of 32 data bytes.

dst array		
	col 0	col 1
row 0	m	
row 1		

src array		
	col 0	col 1
row 0	m	
row 1		

Problem 7. （10 points）

Consider the following code running on a 32-bit Linux system. The executable object file a.out is compiled and linked using the command

```
unix> gcc -o a.out main.c foo.c
```

and the files `main.c` and `foo.c` consist of the following code:

本地符号

```
/* main.c */
nclude <stdio.h>

static int a = 1;
int b = 2;
int c;

int main()
{
    int c = 3;
    foo();
    {
        int c = 4;
    }
    printf("a=%d, b=%d, c=%d, ", a, b, c);

    short **p = calloc(8, sizeof(char));
    long *a = (long*) (*p + 0x200);
    c = (int) (a + 0x300);
    printf("a=0x%x, c=0x%x\n", a, c);

    return 0;
}
```

全局符号
初始化

全局符号
无初始化

0+0x200*2（short大
小）

全局模块

弱符号

弱符号

```
/* foo.c */
int a, b, c;
void foo()
{
    a = 100;
    b = 200;
    c = 300;
}
```

What is the output of a.out?

a=_____, b=_____, c=_____, a=_____, c=_____

1

200

3局部变量

Problem 8. （10 points）

Consider the C program below. (For space reasons, we are not checking error return codes, so assume that all functions return normally.)

```
int main () {
   if (fork() == 0) {
      if (fork() == 0) {
          printf("3");
      }
      else {
         pid_t pid; int status;
         if ((pid = wait(&status)) > 0) {
            printf("4");
         }
      }
   }
   else {
        printf("2");
        exit(0);
   }
   printf("0");
   return 0;
}
```

For each of the following strings, circle whether (Y) or not (N) this string is a possible output of the program.

A. 32040 Y N
B. 34002 Y N
C. 30402 Y N
D. 23040 Y N
E. 40302 Y N

Problem 9. （12 points）

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.

这道题涉及多级页表的虚拟地址翻译，使用 32 位 Intel 系统，4 KB 页表。

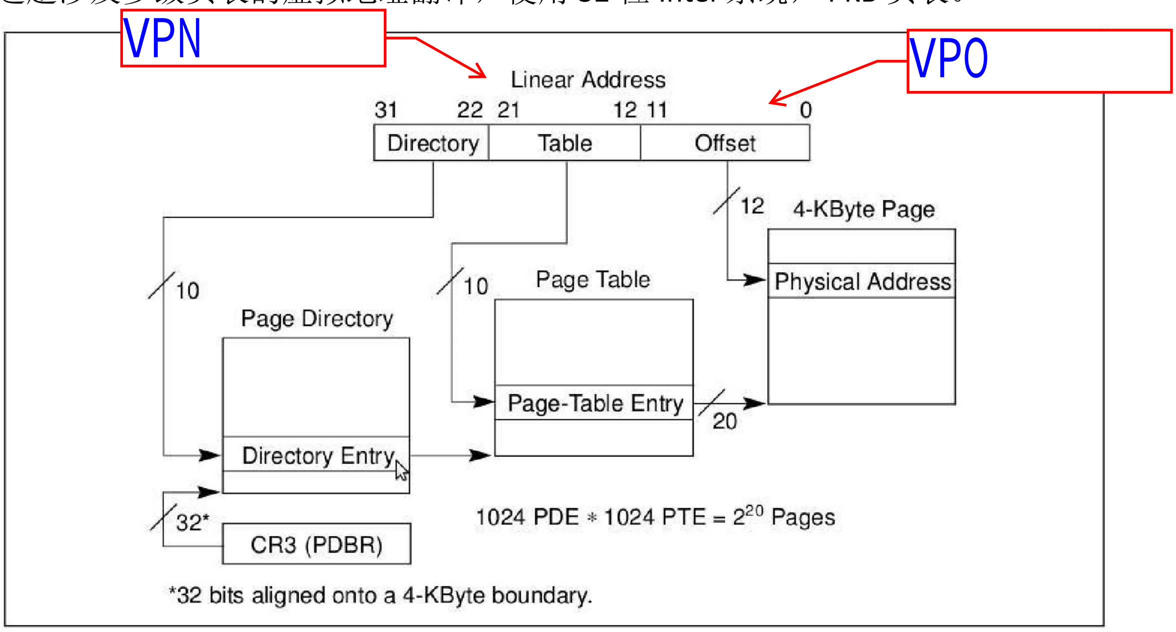

Figure 3-12. Linear Address Translation (4-KByte Pages)

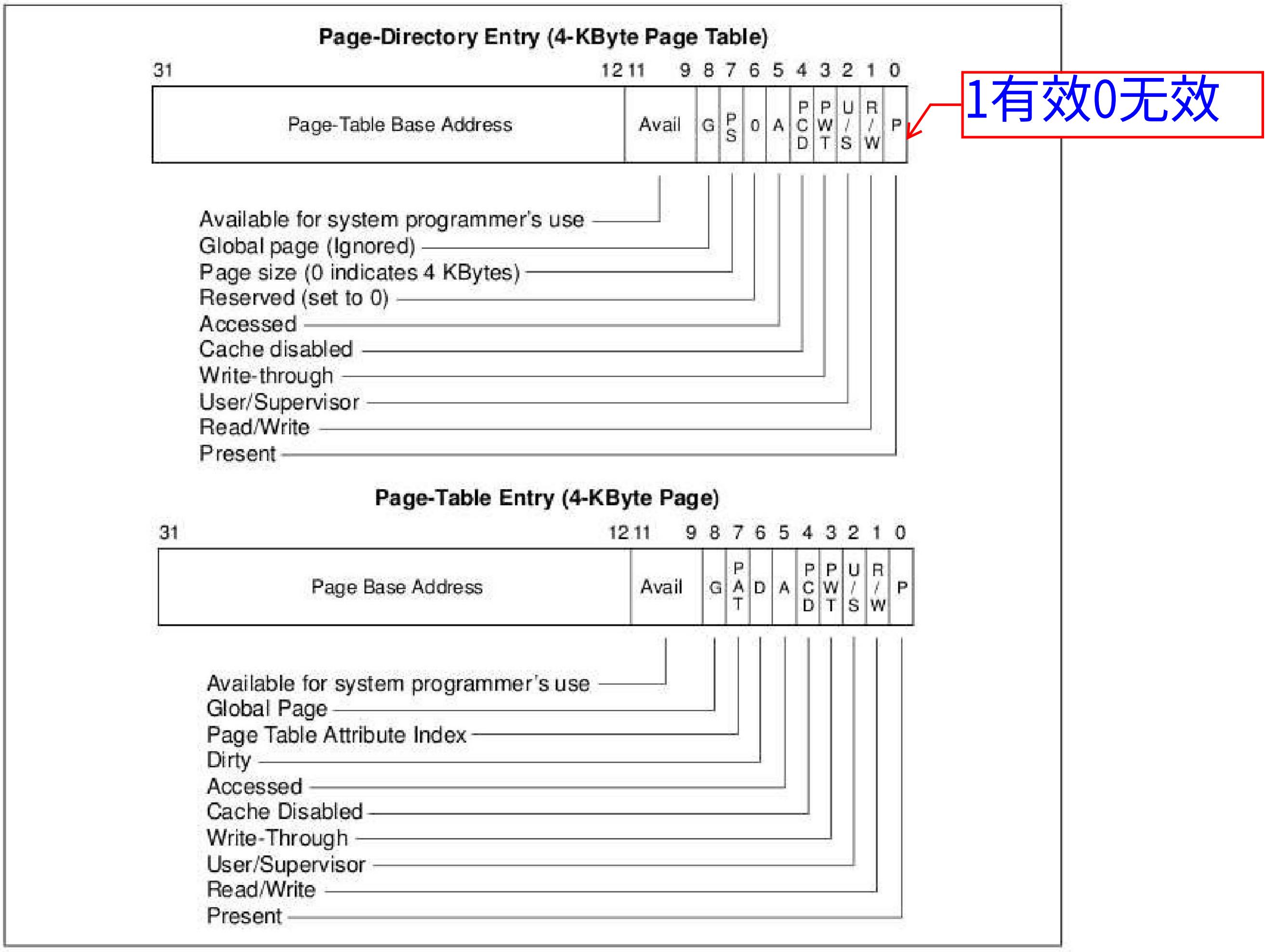

Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

The contents of the relevant sections of memory are shown on this table. All numbers are given in hex- adecimal.Any memory not shown can be assumed to be zero. The Page Directory Base Address is 0x0c23b000.
下表给出了相关部分的存储器内容。所有数字都用十六进制给出。没有显示的存储器假设为零。页目录基地址为 0x0c23b000。

Address	Contents
00023000	beefbee0
00023120	12fdc883
00023200	debcfd23
00023320	d2e52933
00023FFF	bcdeff29
00055002	8974d003
00055004	457bc293
00055008	457bd293
00055464	457be293
0c23b010	01288b52
0c23b020	012aab53
0c23b040	00055d01
0c23b080	0FF2d303
0c23b274	00023d03
0c23b7bc	514d2274
2314d200	0fdc1223
2314d220	d21345a9
2314d4a0	d388bcbd
2314d890	00b32d00
24AEE520	b58cdad1
29DE2504	56ffad02
29DE4400	2ab45cd0
29DE9402	d4732000
29DEE500	1a23cdb0

For each of the following problems, perform the virtual to physical address translation. If an error occurs at any point in the address translation process that would prevent the system from performing the lookup, then indicate this by writing “FAILURE” in (c).
对下面每个问题，进行虚拟地址到物理地址翻译。如果过程中发生错误，则(c) 中写“FAILURE”。

For example, if you were to detect that the present bit in the PDE is set to zero, then you would leave the PTE address in (b) empty, and write “FAILURE”.
例如，如果检测到 PDE 有效位为零，那么(b)中的 PTE 为空，(c)中写“FAILURE”。

1. Read from virtual address **`0x7bcd8001`**:
    a. Physical address of PDE: ____________
    b. Physical address of PTE: ____________
    c. The physical address accessed is ____________

2. Read from virtual address **`0x04002abc`**:
    a. Physical address of PDE: ____________
    b. Physical address of PTE: ____________
    c. The physical address accessed is ____________