学院 专业 班级 学号 姓名

密封线 学生须将答案写在此线以下

鲁东大学 2022—2023 学年第 1 学期

2020 级 软件工程 专业本科卷 B 课程名称 计算机系统基础

课程号（22213332） 考试形式（闭卷笔试） 时间（120 分钟）

题 目	一	二	三	四	五	六	七	总 分	统分人	复核人
得 分										

得分	评卷人

一、信息的表示和处理（本题共 3 小题，满分 20 分）

1、（6 分）考虑下面的 C 函数：

```
int fun1(unsigned word) {
    return (int)((word <<24)>>24);
}
int fun2(unsigned word) {
    return ((int)word <<24)>>24;
}
```

假设在一个采用补码运算的机器上以 32 位程序来执行这些函数。还假设有符号数值的右移是算术右移，而无符号数值的右移是逻辑右移。

填写下表，说明这些函数对几个示例参数的结果。

w	fun1(w)	fun2(w)
0x87654321		
0x000000C9		
0xEDCBA987		

2、（6 分）写一个 C 表达式， 在下列描述的条件下产生 1， 而在其他情况下得到 0。假设 x 是 int 类型。代码应该遵循位级整数编码规则，另外还有一个限制，你不能使用相等(==)和不相等(!=)测试。

A. x 的任何位都等于 1。

B. x 的最低有效字节中的位都等于 1。

C. x 的最高有效字节中的位都等于 0。

3、（8 分）假设我们在对有符号值使用补码运算的 32 位机器上运行代码。对于有符号值使用的是算术右移，而对于无符号值使用的是逻辑右移。变量的声明和初始化如下：

```
int x = foo() ; /* Arbitrary value */
int y = bar() ; /* Arbitrary value */
unsigned ux = x;
unsigned uy = y;
```

判断下面每个 C 表达式的值为真(等于 1)还是为假(等于 0)。

```
A.(x>0)||(x-1<0)              真      假
B.(x&7)!=7 ||(x<<29<0)        真      假
C.(x*x)>=0                    真      假
D.x<0 || -x<=0                真      假
E.x>0 || -x>=0                真      假
F.x-ux == uy-y                真      假
G.x*~y + uy*ux == -x          真      假
H.x+uy == ux+y                真      假
```

得分	评卷人

二、程序的机器级表示（本题共 3 小题，满分 30 分）

1、（14 分）C 代码开始的形式如下：

```
long test(long x, long y) {
    long val = ________;
    if (_______) {
        if (_______)
            val = ________;
        else
            val = ________;
    } else if (_______)
        val = ________;
    return val;
}
```

GCC 会产生如下汇编代码：

```
  long test(long x, long y)
  x in %rdi, y in %rsi
test:
  leaq    0(, %rdi, 8), %rax
  testq   %rsi, %rsi
  jle     .L2
  movq    %rsi, %rax
```

```
  subq      %rdi, %rax
  movq      %rdi, %rdx
  andq      %rsi, %rdx
  cmpq      %rsi, %rdi
  cmovge    %rdx, %rax
  ret
.L2:
  addq      %rsi, %rdi
  cmpq      $-2, %rs
  cmovle    %rdi, %rax
  ret
```

填补 C 代码中缺失的表达式。

2、（6 分）考虑下面的 C 代码，其中 H 和 J 是用#Define 声明的常量。

```
int array1[H][J];
int array2[J][H];
void copy_array(int x, int y) {
    array2[y][x] = array1[x][y];
}
```

假设上述 C 代码生成以下 x86-64 汇编代码：

```
# On entry:
#     %edi = x
#     %esi = y
#
copy_array:
      movslq  %edi, %rdi
      movslq  %esi, %rsi
      movq    %rsi, %rdx
      salq    $4, %rdx
      subq    %rsi, %rdx
      addq    %rdi, %rdx
      leaq    0(,%rdi,8), %rax
      subq    %rdi, %rax
      addq    %rsi, %rax
      movl    array1(,%rax,4), %eax
      movl    %eax, array2(,%rdx,4)
      ret
```

问：H 和 J 的值是多少？ 解答：H =　　J =

3、（10 分）在下面这个过程中，去掉了 switch 语句的主体：

```
1    long switch_prob(long x, long n) {
2        long result = x;
3        switch(n) {
4            /*Fill in code here*/
5
6        }
7        return result;
8    }
```


下图给出了这个过程的反汇编机器代码。

```
     long switch_prob(long x, long n)
     x in %rdi, n in %rsi
1    0000000000400590 <switch_prob>:
2      400590:  48 83 ee 3c             sub    $0x3c,%rsi
3      400594:  48 83 fe 05             cmp    $0x5,%rsi
4      400598:  77 29                   ja     4005c3 <switch_prob+0x33>
5      40059a:  ff 24 f5 f8 06 40 00    jmpq   *0x4006f8(,%rsi,8)
6      4005a1:  48 8d 04 fd 00 00 00    lea    0x0(,%rdi,8),%rax
7      4005a8:  00
8      4005a9:  c3                      retq
9      4005aa:  48 89 f8                mov    %rdi,%rax
10     4005ad:  48 c1 f8 03             sar    $0x3,%rax
11     4005b1:  c3                      retq
12     4005b2:  48 89 f8                mov    %rdi,%rax
13     4005b5:  48 c1 e0 04             shl    $0x4,%rax
14     4005b9:  48 29 f8                sub    %rdi,%rax
15     4005bc:  48 89 c7                mov    %rax,%rdi
16     4005bf:  48 0f af ff             imul   %rdi,%rdi
17     4005c3:  48 8d 47 4b             lea    0x4b(%rdi),%rax
18     4005c7:  c3                      retq
```

跳转表驻留在内存的不同区域中。可以从第 5 行的间接跳转看出来，跳转表的起始地址为 0x4006f8。用调试器 GDB， 我们可以用命令 x/6gx 0x4006f8 来检查组成跳转表的 6 个 8 字节字的内存。GDB 打印出下面的内容：

```
(gdb) x/6gx 0x4006f8
0x4006f8:    0x00000000004005a1    0x00000000004005c3
0x400708:    0x00000000004005a1    0x00000000004005aa
0x400718:    0x00000000004005b2    0x00000000004005bf
```

用 C 代码填写开关语句的主体，使它的行为与机器代码一致。

得分	评卷人

三、优化程序性能（本题共 1 小题，满分 6 分）

假设写一个对多项式求值的函数，这里多项式的次数为 n，系数为 a_0，a_1，…，a_n。对于值 x，我们对多项式求值，计算

$$a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$$

这个求值可以用下面的函数来实现，参数包括一个系数数组 a、值 x 和多项式的次数 degree(上面等式中的值 n)。在这个函数的一个循环中，我们计算连续的等式的项，以及连续的 x 的幂：

```
1  double poly(double a[], double x, long degree)
2  {
3     long i;
4     double result = a[0];
5     double xpwr = x; /* Equals x^i at start of loop */
6     for (i = 1; i <= degree; i++) {
7        result += a[i] * xpwr;
8        xpwr = x * xpwr;
9     }
10    return result;
11 }
```

A.对于次数 n，这段代码执行多少次加法和多少次乘法运算？

B.在我们的参考机上，算术运算的延迟如下图所示，我们测量了这个函数的 CPE 等于 5.00。根据由于实现函数第 7~8 行的操作迭代之间形成的数据相关，解释为什么会得到这样的 CPE。

运算	整数			浮点数		
	延迟	发射	容量	延迟	发射	容量
加法	1	1	4	3	1	1
乘法	3	1	1	5	1	2
除法	3 ~ 30	3 ~ 30	1	3 ~ 15	3 ~ 15	1

得分	评卷人

四、存储器层次结构（本题共 1 小题，满分 20 分）

我们考虑一个 128 字节的数据高速缓存，它是 2 路组相联的，每个高速缓存行中可以容纳 4 个 double。假定 double 需要 8 个字节。

对于下面的代码，我们假设一个冷缓存。此外，我们考虑缓存对齐的 32 双精度数组 A（即，A[0] 被加载到第一组缓存行的第一个槽中）。所有其他变量都保存在寄存器中。该代码由满足 m*n = 32 的正整数 m 和 n 参数化（即，如果你知道一个，你就会知道另一个）。

回想一下，未命中率定义为 #misses/#accesses。

```
float A[32], t = 0;
for(int i = 0; i < m; i++)
   for(int j = 0; j < n; j++)
      t += A[j*m + i];
```

回答以下问题：

A. 缓存可以容纳多少个 double?

B. 缓存有多少组?

C. 对于 m = 1:

(a) 确定未命中率。

(b) 发生了什么未命中?

(c) 代码是否具有关于访问 A 和此缓存的时间局部性?

D. 对于 m = 2:

(a) 确定未命中率。

(b) 发生了什么未命中?

E. 对于 m = 16:

(a) 确定未命中率。

(b) 发生了什么未命中?

(c) 代码是否具有关于访问 A 和此缓存的空间局部性?

得分	评卷人

五、链接（本题共 1 小题，满分 6 分）

考虑可执行目标文件 a.out，它是使用命令

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

编译和链接的，文件 main.c 和 foo.c 由以下代码组成：

```
/* main.c */
#include <stdio.h>

int a = 9, b = 8;
double c;

void foo();

int main()
{
    foo();
    printf("a=%d b=%d c=%lf\n", a, b, c);
    return 0;
}
```

```
/* foo.c */
int a;
static int b;
double c = 7;

void foo()
{
    a = 6;
    b = 5;
    c = 4;
}
```

问：a.out 的输出是什么？

得分	评卷人

六、异常控制流（本题共 1 小题，满分 6 分）

考虑以下 C 程序。（由于篇幅原因，我们不检查错误返回码，假设所有函数都正常返回。）

```
int main()
{
    int val = 2;

    printf("%d", 0);
    fflush(stdout);

    if (fork() == 0) {
        val++;
        printf("%d", val);
        fflush(stdout);
    }
    else {
        val--;
        printf("%d", val);
        fflush(stdout);
        wait(NULL);
    }
    val++;
    printf("%d", val);
    fflush(stdout);
    exit(0);
}
```

对于以下每个字符串，圈出 (Y) 或 (N) 该字符串是否是程序的可能输出。

A. 01342 Y N
B. 01234 Y N
C. 03412 Y N

得分	评卷人

七、虚拟存储（本题共 1 小题，满分 12 分）地址翻译。

这个问题涉及将虚拟地址转换为物理地址的方式。

一个系统具有以下参数：

- 虚拟地址为 20 位宽。
- 物理地址为 18 位宽。
- 页大小为 1024 字节。
- TLB 是 2 路组相联的，总共有 16 个条目。

TLB 的内容和页表的前 32 个条目如下所示。所有数字都是以十六进制给出。

TLB

Index	Tag	PPN	Valid
0	03	C3	1
	01	71	0
1	00	28	1
	01	35	1
2	02	68	1
	3A	F1	0
3	03	12	1
	02	30	1
4	7F	05	0
	01	A1	0
5	00	53	1
	03	4E	1
6	1B	34	0
	00	1F	1
7	03	38	1
	32	09	0

Page Table

VPN	PPN	Valid	VPN	PPN	Valid
000	71	1	010	60	0
001	28	1	011	57	0
002	93	1	012	68	1
003	AB	0	013	30	1
004	D6	0	014	0D	0
005	53	1	015	2B	0
006	1F	1	016	9F	0
007	80	1	017	62	0
008	02	0	018	C3	1
009	35	1	019	04	0
00A	41	0	01A	F1	1
00B	86	1	01B	12	1
00C	A1	1	01C	30	0
00D	D5	1	01D	4E	1
00E	8E	0	01E	57	1
00F	D4	0	01F	38	1

第一部分：

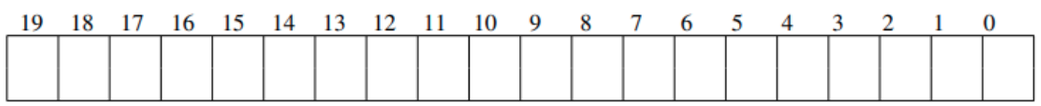
1. 下图显示了虚拟地址的格式。请在图表中标注以下字段：

   VPO　虚拟页偏移量

   VPN　虚拟页号

   TLBI　TLB 索引

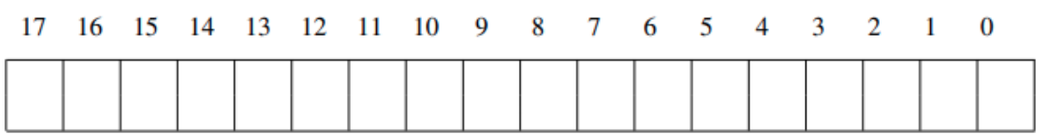
   TLBT　TLB 标记



2. 下图显示了物理地址的格式。请在图表中标注以下字段：

   PPO　物理页偏移量

   PPN　物理页号

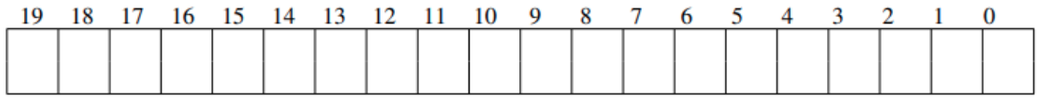


第二部分：

对于给定的虚拟地址，请指明访问的 TLB 条目和物理地址。指示 TLB 是否未命中以及是否发生页面错误。如果出现页面错误，请为“PPN”输入“-”并将物理地址留空。

虚拟地址：078E6

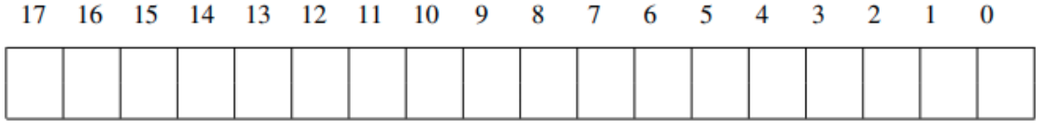
1. 虚拟地址(每框一位)



2. 地址翻译

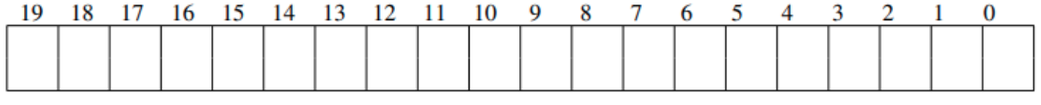
参数	值	参数	值
VPN	0x	TLB 命中? (Y/N)	
TLB Index	0x	缺页? (Y/N)	
TLB Tag	0x	PPN	0x

3. 物理地址(每框一位)



虚拟地址：04AA4

1. 虚拟地址(每框一位)



2. 地址翻译

参数	值	参数	值
VPN	0x	TLB 命中? (Y/N)	
TLB Index	0x	缺页? (Y/N)	
TLB Tag	0x	PPN	0x

3. 物理地址(每框一位)