提醒：请诚信应考，考试违规将带来严重后果！

教务处填写：

___年___月___日
考 试 用

湖南大学课程考试试卷

课程名称： 计算机系统 ；**课程编码：** CS04033 ；

试卷编号： A ；**考试形式：** 闭卷 ；**考试时间：** **120** 分钟。

题 号	一	二	三	四	五	六	七	八	九	十	总分
应得分	10	15	10	15	10	20					100
实得分											
评卷人											

湖南大学课程考试试卷

专业班级：

学号：

姓名：

湖南大学教务处

装订线（题目不得超过此线）

（请在答题纸内作答！）

1.简答题（10 分）

在某机器中，浮点数采用 IEEE 754 标准表示，每个浮点数都对应着一个二进制编码，去掉负数（即符号位为 1 的那些编码）、去掉特殊值（即阶码位全 1 的那些编码），剩下的编码按照这个二进制编码对应的**整数值**排序并从 0 开始编号，第 k 个其对应的浮点数值记为f_k，例如全 0 编码 0000……000 的序号是 0，$f_0 = 0$, 并记最大的编号为max。***显然***，越排在后面的编码，对应的浮点数值越大；同时将相邻两个编码的浮点数差值定义为步长，即 $step_i = f_{i+1} - f_i$，***显然***，步长值是单调非减的。

（1） 请证明第一个显然，即如果有 $0 \le m < n \le max$, 则有$f_m < f_n$。（5 分）

（2） 请证明第二个显然，即如果有 $0 \le m < n < max$, 则有$step_m \le step_n$。（5 分）

2.程序填空题（15 分，每空 3 分）

如下是一个 c 语言程序及其对应的汇编代码（32 位机，小端环境下编译），请参照汇编代码，完成 c 程序的空缺部分。

c 语言程序：

```
#include <stdio.h>
#define X  16
#define Y     (1)
int array1[X][Y];
int array2[X];

int test()
{
    int sum=    (2)    ;
    int i=0,j=0,k=0;
    for(i=1,j=2;i<X && j<Y;i++)
    {
```

```
    k=   (3)   ;
        array2[i]+=   (4)   ;
        j++;
    }
    return   (5)   ;
}
int main()
{
    return 0;
}
```

汇编代码如下：

```
test:
    pushl  %ebp
    movl   %esp, %ebp
    pushl  %ebx
    subl   $16, %esp
    movl   $6, -12(%ebp)
    movl   $0, -20(%ebp)
    movl   $0, -16(%ebp)
    movl   $0, -8(%ebp)
    movl   $1, -20(%ebp)
    movl   $2, -16(%ebp)
    jmp .L2
    .cfi_offset 3, -12
.L6:
    cmpl   $15, -16(%ebp)
    jg  .L3
    movl   -16(%ebp), %eax
    subl   $15, %eax
    jmp .L4
.L3:
    movl   $0, %eax
.L4:
    movl   %eax, -8(%ebp)
    movl   -20(%ebp), %eax
    movl   array2(,%eax,4), %edx
    movl   -20(%ebp), %eax
    imull  $23, %eax, %eax
    addl   -16(%ebp), %eax
    movl   array1(,%eax,4), %ecx
    movl   -8(%ebp), %eax
    movl   array2(,%eax,4), %ebx
    movl   $0, %eax
    subl   %ebx, %eax
    addl   %eax, %eax
```

```
    addl   %ecx, %eax
    addl   %eax, %edx
    movl   -20(%ebp), %eax
    movl   %edx, array2(,%eax,4)
    addl   $1, -16(%ebp)
    addl   $1, -20(%ebp)
.L2:
    cmpl   $15, -20(%ebp)
    jg .L5
    cmpl   $22, -16(%ebp)
    jle .L6
.L5:
    movl   array2, %eax
    movl   %eax, %edx
    addl   -12(%ebp), %edx
    movl   array1, %eax
    addl   %edx, %eax
    addl   $16, %esp
    popl   %ebx
    popl   %ebp
    ret
```


3.程序填空题（10 分，第 1，2 空每空 3 分，每 3 空 4 分）

如下是一个 c 语言程序及其对应的汇编代码（32 位机，小端环境下编译），请参照汇编代码，完成 c 程序的空缺部分。

提示：

```
#include <stdio.h>
union u
{
    int    i;
    int    *pi;
    char   *pc;
}ua;
int a[3]={0,1,2};
int test1(int *x,int y)
{
    int result =0;
    if(__(1)_____)
        result=2;
    return result;

}
int test2(int x,int y)
{
    int result=0;
```

```
if(   (2)    )
        result=    (3)    ;
    return result;
}
int main()
{
    int i=0;
    ua.pi=&i;
    int b=ua.i;
    i=test2(20,b+10);
}
```

汇编代码如下：

```
a:
    .long  0
    .long  1
    .long  2
test1:
    pushl  %ebp
    movl   %esp, %ebp
    subl   $16, %esp
    movl   $0, -4(%ebp)
    movl   8(%ebp), %eax
    movl   (%eax), %eax
    movl   12(%ebp), %edx
    sall   $2, %edx
    addl   $a, %edx
    cmpl   %edx, %eax
    jbe .L2
    movl   $2, -4(%ebp)
.L2:
    movl   -4(%ebp), %eax
    leave
    ret

test2:
    pushl  %ebp
    movl   %esp, %ebp
    subl   $24, %esp
    movl   $0, -4(%ebp)
    movl   8(%ebp), %eax
    movl   %eax, %edx
    imull  12(%ebp), %edx
    movl   ua, %eax
    cmpl   %eax, %edx
    jle .L4
```

```
    movl   ua, %eax
    leal   10(%eax), %edx
    movl   ua, %eax
    movl   %edx, 4(%esp)
    movl   %eax, (%esp)
    call   test1
    movl   ua, %edx
    movl   %eax, 4(%esp)
    movl   %edx, (%esp)
    call   test1
    movl   %eax, -4(%ebp)
.L4:
    movl   -4(%ebp), %eax
    leave
    ret

main:
    pushl  %ebp
    movl   %esp, %ebp
    subl   $24, %esp
    movl   $0, -8(%ebp)
    leal   -8(%ebp), %eax
    movl   %eax, ua
    movl   ua, %eax
    movl   %eax, -4(%ebp)
    movl   -4(%ebp), %eax
    addl   $10, %eax
    movl   %eax, 4(%esp)
    movl   $20, (%esp)
    call   test2
    movl   %eax, -8(%ebp)
    movl   -8(%ebp), %eax
    leave
    ret
```


4.综合应用题（15 分）

小明写了一个验证冰雹猜想（又称角谷猜想）的程序，即任取一个自然数 x，如果是奇数就乘以 3 再加 1，如果是偶数就除 2，这样经过若干次，最终会得到 1。

其代码如下：

```
#include <stdio.h>
int P(int x)
{
    printf("%d\n",x);
    if(x<0 || x==1) return 0;
```

```
if(x%2==0) return P(x/2);
    return P(x*3+1);
}
int main()
{
    P(5);
    return 0;
}
```

运行后，会依次打印出 5，16，8，4，2，1。

在 32 位机，小端环境下编译链接，得到的汇编代码如下：

```
080483e4 <P>:
 80483e4: 55                    push   %ebp
 80483e5: 89 e5                 mov    %esp,%ebp
 80483e7: 83 ec 18              sub    $0x18,%esp
 80483ea: b8 40 85 04 08        mov    $0x8048540,%eax
 80483ef: 8b 55 08              mov    0x8(%ebp),%edx
 80483f2: 89 54 24 04           mov    %edx,0x4(%esp)
 80483f6: 89 04 24              mov    %eax,(%esp)
 80483f9: e8 02 ff ff ff        call   8048300 <printf@plt>
 80483fe: 83 7d 08 00           cmpl   $0x0,0x8(%ebp)
 8048402: 78 06                 js     804840a <P+0x26>
 8048404: 83 7d 08 01           cmpl   $0x1,0x8(%ebp)
 8048408: 75 07                 jne    8048411 <P+0x2d>
 804840a: b8 00 00 00 00        mov    $0x0,%eax
 804840f: eb 34                 jmp    8048445 <P+0x61>
 8048411: 8b 45 08              mov    0x8(%ebp),%eax
 8048414: 83 e0 01              and    $0x1,%eax
 8048417: 85 c0                 test   %eax,%eax
 8048419: 75 16                 jne    8048431 <P+0x4d>
 804841b: 8b 45 08              mov    0x8(%ebp),%eax
 804841e: 89 c2                 mov    %eax,%edx
 8048420: c1 ea 1f              shr    $0x1f,%edx
 8048423: 01 d0                 add    %edx,%eax
 8048425: d1 f8                 sar    %eax
 8048427: 89 04 24              mov    %eax,(%esp)
 804842a: e8 b5 ff ff ff        call   80483e4 <P>
 804842f: eb 14                 jmp    8048445 <P+0x61>
 8048431: 8b 55 08              mov    0x8(%ebp),%edx
 8048434: 89 d0                 mov    %edx,%eax
 8048436: 01 c0                 add    %eax,%eax
 8048438: 01 d0                 add    %edx,%eax
 804843a: 83 c0 01              add    $0x1,%eax
 804843d: 89 04 24              mov    %eax,(%esp)
 8048440: e8 9f ff ff ff        call   80483e4 <P>
```

```
 8048445: c9                    leave
 8048446: c3                    ret

08048447 <main>:
 8048447: 55                    push   %ebp
 8048448: 89 e5                 mov    %esp,%ebp
 804844a: 83 e4 f0              and    $0xfffffff0,%esp
 804844d: 83 ec 10              sub    $0x10,%esp
 8048450: c7 04 24 05 00 00 00    movl   $0x5,(%esp)
 8048457: e8 88 ff ff ff        call   80483e4 <P>
 804845c: b8 00 00 00 00        mov    $0x0,%eax
 8048461: c9                    leave
 8048462: c3                    ret
```

（1）假设在刚开始进入 main 程序代码时，即正准备执行 0x8048447 处的 push %ebp 时，%esp 的值为 0xbffff1fc，则在调用 P(4)，刚进入 P 函数，即正准备执行 0x80483e4 处的 push %ebp 时，%esp，%ebp 的值分别是？内存中%esp 处和%esp+4 处的四个字节值分别是？（每空 2 分，共 8 分）

（2）假设在某机器上，int 为 128 个 bit，在对于某些整数进行验证时，由于步骤较多，会导致栈空间耗尽而提示“段错误”。请改写函数 P，与当前函数 P 的功能相同，但不会出现此类问题，并分析不会出现此类问题的原因。（7 分）

5.综合设计题（10 分）

在本学期，我们学习了相对简单 CPU，其指令格式如下：

NOP 0000 0000	**LDAC** 0000 0001	**STAC** 0000 0010	**MVAC** 0000 0011
MOVR 0000 0100	**JUMP** 0000 0101	**JMPZ** 0000 0110	**JPNZ** 0000 0111
ADD 0000 1000	**SUB** 0000 1001	**INAC** 0000 1010	**CLAC** 0000 1011
AND 0000 1100	**OR** 0000 1101	**XOR** 0000 1110	**NOT** 0000 1111

其对应的状态图及数据通路图如图 5.1 和图 5.2 所示：

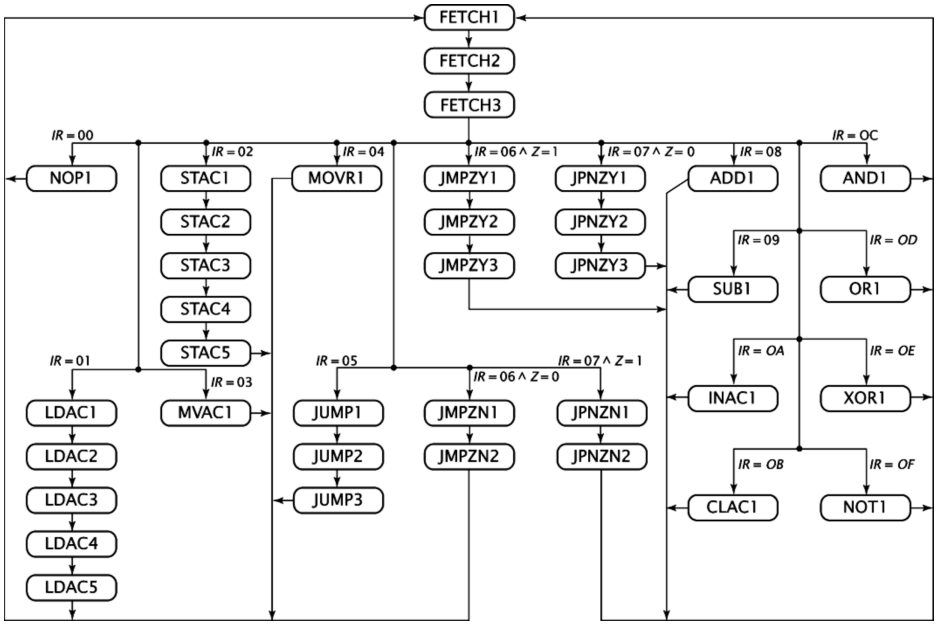

图 5.1 相对简单 CPU 状态图

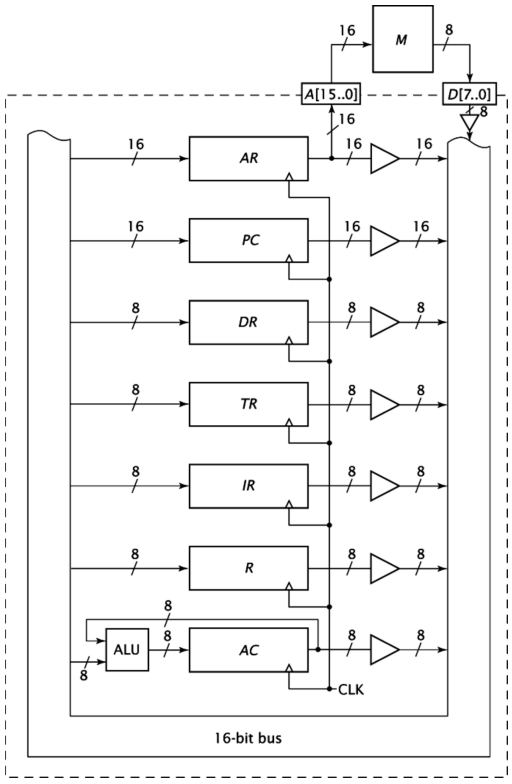

图 5.2 相对简单 CPU 数据通路图

现需要修改相对简单 CPU，使之包含一个新的 8 位的寄存器 B 及下面 3 条新的指令。

指令	指令码	操作
ADDB	0001 1000	AC ← AC + B
SUBB	0001 1001	AC ← AC - B
ANDB	0001 1100	AC ← AC Λ B

1）请给出修改后的状态图（可以只画出增加的关键部分）（4 分）

2）请给出对与数据通路中的 ALU 和寄存器部分的修改（可以只画出增加或修改的关键部分）（6 分）

6．典型的 ELF 可重定位目标文件格式如下图 6.1 所示，请问：

（1） 与待链接的可重定位目标文件相比，链接后的可执行文件目标文件的内容有哪些变化？为什么会有这些变化？（4 分）

（2） 如果待链接的两个目标文件，分别来源于如图 5.2 与图 5.3 中的两段 C 代码，请问可能会产生什么问题？为什么？（4 分）

（3） 对于图 6.2 与图 6.3 的代码进行链接操作形成可执行文件 P 时，可能会发生哪些重定位？并简述这些重定位的基本过程。（4 分）

（4） 请从产生新进程继而加载执行它的角度，描述在终端使用“./P”到屏幕输出相应运行结果的实际过程。（4 分）

（5） 一旦可执行文件 P 运行完毕，直接会导致产生什么信号？信号的接收者是谁？对于信号的接收者而言，信号不排队是什么意思？（4 分）

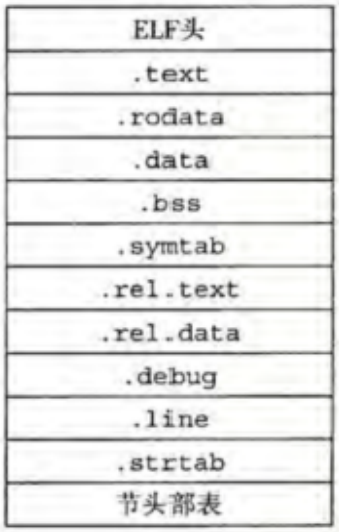

图 6.1

```
//main11.c
#include <stdio.h>
void f(void);
int x;
int main() {
x = 15213;
y = 0;
 f();
printf("x = %d\n",
x);
return 0; }
```

图 6.2

```
//f11.c
double x;
int f() {
x = 1.0;
}
```

图 6.3

7. （20 分）某 32 位的 Intel 系统，使用 2 级页表处理虚拟内存地址转换，页表大小为 4K，假设所有进程在特权模式下运行。内存片段内容如下表，其中所有数字都是**十六进制**。任何未显示的内存都假定为零。所有的页表都是 4K 对齐的，页目录的基址（PDBA）是：0x0045d000。

对于以下给出的 3 个虚拟地址，请执行虚拟地址到物理地址的转换。如果在地址转换过程中发生错误，请找到导致失败的页表条目的物理地址。如果一级页表 PDE 中的有效位(P)设置为零，那么二级页表 PTE 地址填写（不在内存）。

(提示：在本 32 位系统中，虚拟地址和物理地址都是 32 位；每个页表条目 4 个字节，32 位页表条目中高 20 位表示页表基址，低 12 位为权限位，最低位为有效位，如图 7.1 和图 7.2)

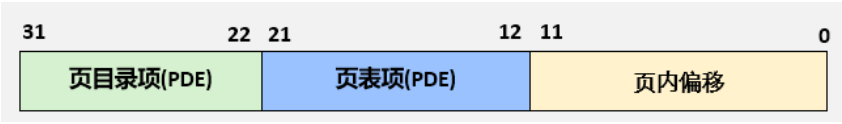

图 7.1. 线性地址结构

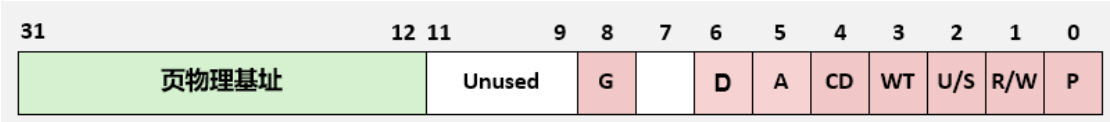

图 7.2. 页目录或页表项结构

P: Child page (table) is present in memory (1) or not (0)

表 7.1. TLB

TLB			
Index	Tag	PPN	Valid
0	0x03506	0x98f8a	1
	0x27f4a	0x34abe	0
1	0x1f7ee	0x95cbc	0
	0x2a064	0x72954	1
2	0x1f7f0	0x95ede	0
	0x2005d	0xaa402	0
3	0x3fc2e	0x2029e	1
	0x3df82	0xff644	0

表 7.2 内存片段

Address	Contents
000c3020	345ab236
000c3080	345ab237
000c332f	08e4523f
000c3400	93c2ed98
000c3cbc	34abd237
000c3ff0	93c2ed99
000c4020	8e56e237
000c432f	33345237
000c4400	43457292
000c4cbc	385ed293
000c4ff0	c3726292
0045d000	000c3292
0045d028	000c4297

Address	Contents
0045d032	0df2a292
0045d0a0	000c3297
0045d3ff	0df2a236
0045d9fc	0df2a237
0df2a000	deded000
0df2a080	bc3de239
0df2a3fc	000c4296
0df2a4a0	00324236
0df2a4fc	df72c9a6
0df2b080	01f008c3
0df2bff0	000c5112

对于虚拟地址 1：0x9fd28c10

A1）虚拟地址的 TLBT 为：0x_____，TLB 命中情况（命中/不命中）：_____

A2）如果 TLB 命中，转换后的物理地址是：0x_____

如果 TLB 不命中，请继续回答（3）和（4）

A3） PDE 的物理地址：0x_____，PTE 的物理地址：0x_____

A4）地址转换情况（成功/失败）：_____

如果成功，转换后的物理地址为：0x_____

如果失败，导致失败的页表条目的物理地址为：0x_____

对于虚拟地址 2：0x0a32fcd0

B1）虚拟地址的 TLBT 为：0x_____，TLB 命中情况（命中/不命中）：_____

B2）如果 TLB 命中，转换后的物理地址是：0x_____

如果 TLB 不命中，请继续回答（3）和（4）

B3） PDE 的物理地址：0x_____，PTE 的物理地址：0x_____

B4）地址转换情况（成功/失败）：_____

如果成功，转换后的物理地址为：0x_____

如果失败，导致失败的页表条目的物理地址为：0x_____

对于虚拟地址 3：0x0d4182c0

C1）虚拟地址的 TLBT 为：0x_____，TLB 命中情况（命中/不命中）：_____

C2）如果 TLB 命中，转换后的物理地址是：0x_____

如果 TLB 不命中，请继续回答（3）和（4）

C3） PDE 的物理地址：0x_____，PTE 的物理地址：0x_____

C4）地址转换情况（成功/失败）：_____

如果成功，转换后的物理地址为：0x_____

如果失败，导致失败的页表条目的物理地址为：0x_____