哈尔滨工业大学（深圳）2021 年夏季学期

主管领导审核签字

计算机系统试题（A）

题号	一	二	三	四	五	总分
得分						
阅卷人						

考生须知：本次考试为闭卷考试，考试时间为 120 分钟，满分 100 分。

授课教师 密 姓名 封 学号 线 院系

一、 单项选择题（每小题 2 分，共 30 分）

1. 对于 IEEE 浮点数，如果减少 1 位指数位，将其用于小数部分，将会有怎样的效果？答：（ ）
   A. 能表示更多数量的实数值，但实数值取值范围比原来小了。
   B. 能表示的实数数量没有变化，但数值的精度更高了。
   C. 能表示的最大实数变小，最小的实数变大，但数值的精度更高。
   D. 以上说法都不正确。

2. 假设有下面 x 和 y 的程序定义 int x = a >> 2; int y = (x + a) / 4; 那么有多少个位于闭区间[-8,8]的整数 a 能使得 x 和 y 相等？（ ）
   A.12 B.13 C.14 D.15

3. 给定一个实数，会因为该实数表示成单精度浮点数而发生误差。不考虑 NaN 和 Inf 的情况，该绝对误差的最大值为（ ）（单精度浮点数的阶码 8 位，尾数 23 位）
   A. 2^103 B. 2^104 C. 2^230 D. 2^231

4. 已知下面的数据结构，假设在 X86-64 环境下要求对齐，这个结构的总的大小是多少个字节？如果重新排列其中的字段，最少可以达到多少个字节？（ ）

```
struct {
    char a;
    float *b;
    long c;
    short d;
    int e;
    };
```

   A.32, 24 B.24, 24 C.28, 26 D.32, 26

5. x86-64 体系结构中，下面哪个说法是不正确的？（ ）
   A.leal 指令可以用来计算乘法表达式的值
   B.x86_64 机器可以使用栈来给函数传递参数
   C.在一个函数内，改变任一寄存器的值之前必须先将其原始数据保存在栈内
   D.程序中的整数常量在编译阶段被初始化

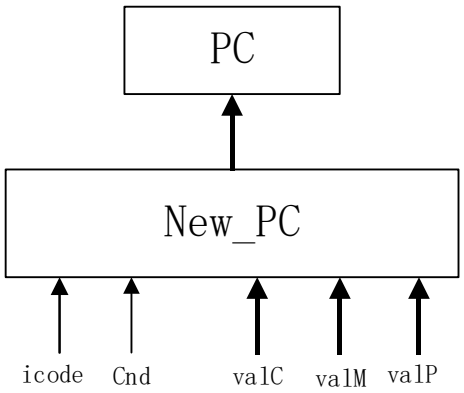
6. 下列的指令组中，哪一组指令只改变条件码，而不改变寄存器的值？（ ）
   A. CMP, SUB
   B. TEST, AND
   C. CMP, TEST
   D. LEAL, CMP

7. 下列哪项不是 Y86-64 流水线 CPU 中的冒险的种类（ ）
   A. 流量冒险　　B.数据冒险　　C.控制冒险　　D.加载使用冒险

8. 关于流水线技术的描述，错误的是（ ）
   A. 流水线技术能够提高执行指令的吞吐率，但也同时增加单条指令的执行时间。
   B. 减少流水线的级数，能够减少数据冒险发生的几率。
   C. 指令间数据相关引发的数据冒险，都可以通过 data forwarding 来解决。
   D. 现代处理器支持一个时钟内取指、执行多条指令，会增加控制冒险的开销。

9. 在 Y86 的 SEQ 实现中，PC（Program Counter，程序计数器）更新的逻辑结构如下图所示，请根据 HCL 描述为①②③④选择正确的数据来源。其中：Icode 为指令类型，Cnd 为条件是否成立，valC 表示指令中的常数值，valM 表示来自返回栈的数据，valP 表示 PC 自增。（ ）

```
Int new_PC = [
    # Call
    Icode = ICALL:  ①  ;
    # Taken branch.
    Icode = IJXX && Cnd:  ②  ;
    # Completion of RET instruction.
    Icode = IRET:  ③  ;
    # Default.
    1:  ④  ;
```



   A. valC, valM, valP, valP　　B. valC, valC, valP, valP
   C. valC, valC, valM, valP　　D. valM, valC, valC, valP

10. 关于信号的描述，以下不正确的是哪一个？（ ）
    A．在任何时刻，一种类型至多只会有一个待处理信号
    B．信号既可以发送给一个进程，也可以发送给一个进程组
    C．SIGTERM 和 SIGKILL 信号既不能被捕获，也不能被忽略
    D．当进程在前台运行时，键入 Ctrl-C，内核就会发送一个 SIGINT 信号给这个前台进程

11. 考虑如下代码，假设 result.txt 的初始内容是“123”。(　　)

```
int main(int argc, char** argv)
{
    int fd = open("result.txt", O_RDWR);
    char str[] = "abc";
    char c;
    write(fd, str, 1);
    read(fd, &c, 1);
    write(fd, &c, 1);
    return 0;
}
```

在这段代码执行完毕之后，result.txt 的内容是什么？（假设所有的系统调用都会成功）。

A. a21　　B. a22　　C. a13　　D. abb

12. 下列系统 I/O 的说法中，正确的是（　）

A．C 语言中的标准 I/O 函数在不同操作系统中的实现代码一样

B．对于同一个文件描述符，混用 RIO 包中的 rio_readnb 和 rio_readn 两个函数不会造成问题

C．C 语言中的标准 I/O 函数是异步线程安全的

D．使用 I/O 缓冲区可以减少系统调用的次数，从而加快 I/O 的速度

13. 虚拟内存系统中的实际物理地址和虚拟地址之间的关系是（　）

A. 1 对 1

B. 多对 1

C. 1 对多

D. 多对多

14. 关于局部性的描述，不正确的是 （　）

A. 循环通常具有很好的时间局部性

B. 循环通常具有很好的空间局部性

C. 数组通常具有很好的时间局部性

D. 数组通常具有很好的空间局部性

15. 虚拟内存发生缺页的时候，正确的叙述是 （　）

A.当发生虚拟内存缺页的时候，程序会直接退出。

B.缺页产生的中断请求通常由 CPU 产生。

C.当处理程序处理完缺页错误之后，会重新执行引发缺页的命令。

D.当一个程序先后重复执行两次的时候，会在相同的指令位置产生缺页错误。

二、填空题 (每空 2 分，共 10 分)

16. 假定编译器规定 int 和 short 型长度分别为 32 位和 16 位，执行下列语句：unsigned short x = 65530; unsigned int y = x; 得到 y 的机器数为_____。（用 16 进制表示，勿省略前导的 0）

17. 一个 C 语言程序在一台 32 位机器上运行。程序中定义了三个变量 x、y 和 z， 其中 x 和 z 为 int 型，y 为 short 型。当 x=127，y=-9 时，执行赋值语句 z=x+y 后，z 的值是_____（用 16 进制表示，勿省略前导的 0）

18. x86-64 体系结构中，常被用来存放函数返回值的寄存器是_____。

19. Y86-64 中请指出 jne t 指令之后执行的那条指令的地址是_____(16 进制表示，假定指令顺序执行不考虑流水线)。

```
0x000:      xorq %rax,%rax
0x002:      jne   t
…
0x019:    t: irmovq $3, %rdx
0x023:      irmovq $4, %rcx
0x02d:      irmovq $5, %rdx
```

20. 对于一个磁盘，其平均旋转速率是 3600 RPM，平均寻道时间是 5ms，单个磁道上平均扇区数量是 900，则这个磁盘的平均访问时间是 _____ ms (结果保留两位小数）

三、判断对错（每小题 2 分，共 10 分，正确打 √、错误打×）

21. （ ）在 64 位的操作系统中，int y = 0x80000000，则 表达式~y + y == 1 的结果为真。
22. （ ）mov 指令可以使用立即数作为目的操作数。
23. （ ）Y86-64 的顺序结构实现中，寄存器文件读时是作为时序逻辑器件看待。
24. （ ）对于发生了冲突不命中类型的高速缓存而言，可以通过增加高速缓存大小解决不命中问题。
25. （ ）固态磁盘性能优于传统机械磁盘，因为其随机读性能和随机写性能基本一致且均快于机械磁盘。

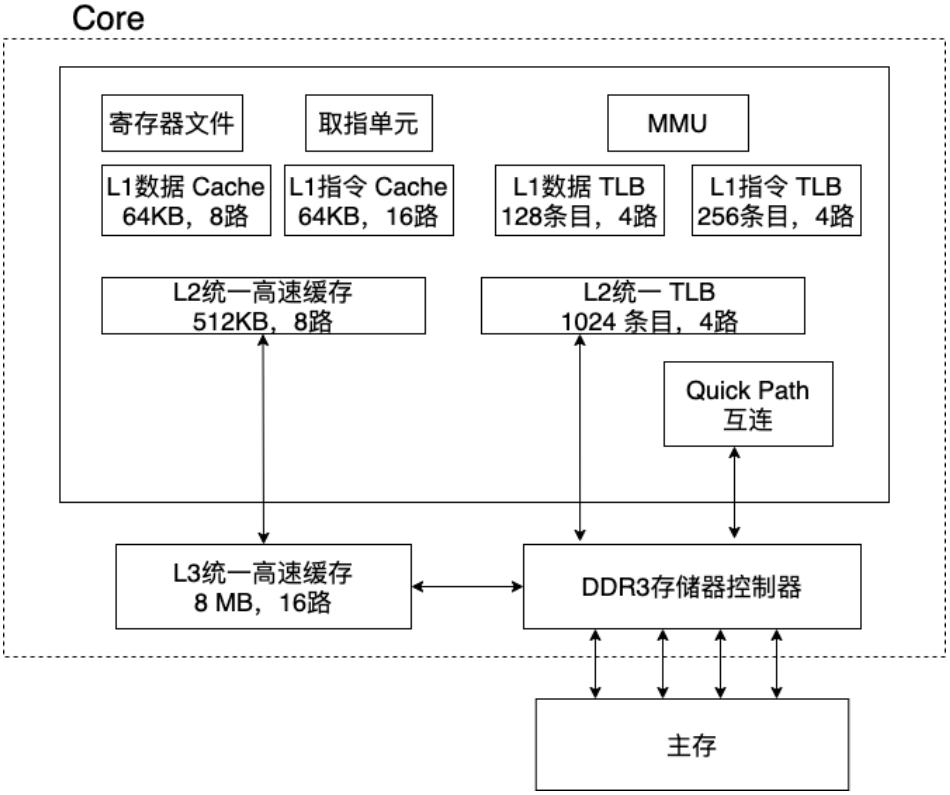
四、系统分析题（30 分）

26. 阅读的 min 函数的汇编结果：（10 分）

```
_Z3minii:
    pushq   %rbp #①
    movq    %rsp, %rbp
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %eax
    cmpl    -8(%rbp), %eax #②
    jl  .L2 #③
    movl    -8(%rbp), %eax
    jmp .L3 #④
.L2:
    movl    -4(%rbp), %eax
.L3:
    popq    %rbp #⑤
    ret
```

解释①-⑤每行指令的功能和作用（写出任意 4 个即可），并解释函数的功能和作用。

27. 某处理器的虚拟地址为 32 位。虚拟内存的页大小是 4KB，物理地址为 48 位，Cache 块大小为 32B。物理内存按照字节寻址。其内部结构如下图所示，依据这个结构，回答下面几个问题。（12 分）



(1)L1 数据 Cache 有多少组，相应的 Tag 位，组索引位和块内偏移位分别是多少？

(2)对于某数据，其访问的虚拟地址为 0x829358B，则该地址对应的 VPO 为多少？对应的 L1 TLBI 位为多少？（用 16 进制表示）

(3)对于某指令，其访问的物理地址为 0x829358B，则该地址访问 L1 Cache 时，CT 位为多少？CO 位为多少？（用 16 进制表示）

28. 两个 C 语言程序 main2.c、addvec.c 如下所示：

```
/* main2.c */
/* $begin main2 */
#include <stdio.h>
#include "vector.h"

int x[2] = {1, 2};
int y[2] = {3, 4};
int z[2];

int main()
{
    addvec(x, y, z, 2);
    printf("z = [%d %d]\n", z[0],
z[1]);
    return 0;
}
/* $end main2 */
```

```
/* addvec.c */
/* $begin addvec */
int addcnt = 0;

void addvec(int *x, int *y,
            int *z, int n)
{
    int i;

    addcnt++;

    for (i = 0; i < n; i++)
        z[i] = x[i] + y[i];
}
/* $end addvec */
```

用如下两条指令编译、链接，生成可执行程序 prog2：

```
gcc -m64 -no-pie -fno-PIC -c addvec.c main2.c
gcc -m64 -no-pie -fno-PIC -o prog2 addvec.o main2.o
```

运行指令 objdump -dxs main2.o 输出的部分内容如下：

```
Disassembly of section .text:

0000000000000000 <main>:
   0:  48 83 ec 08                 sub    $0x8,%rsp
   4:  b9 02 00 00 00              mov    $0x2,%ecx
   9:  ba 00 00 00 00              mov    $0x0,%edx
            a: R_X86_64_32  z
   e:  be 00 00 00 00              mov    $0x0,%esi
            f: R_X86_64_32  y
  13:  bf 00 00 00 00              mov    $0x0,%edi
            14: R_X86_64_32 x
  18:  e8 00 00 00 00              callq  1d <main+0x1d>
            19: R_X86_64_PC32  addvec-0x4
  1d:  8b 0d 00 00 00 00           mov    0x0(%rip),%ecx        # 23 <main+0x23>
            1f: R_X86_64_PC32  z
  23:  8b 15 00 00 00 00           mov    0x0(%rip),%edx        # 29 <main+0x29>
            25: R_X86_64_PC32  z-0x4
  29:  be 00 00 00 00              mov    $0x0,%esi
            2a: R_X86_64_32 .rodata.str1.1
  2e:  bf 01 00 00 00              mov    $0x1,%edi
  33:  b8 00 00 00 00              mov    $0x0,%eax
  38:  e8 00 00 00 00              callq  3d <main+0x3d>
            39: R_X86_64_PC32  __printf_chk-0x4
  3d:  b8 00 00 00 00              mov    $0x0,%eax
  42:  48 83 c4 08                 add    $0x8,%rsp
  46:  c3                          retq
```

objdump -dxs prog2 输出的部分内容如下（■是没有显示的隐藏内容）：

```
SYMBOL TABLE:
0000000000400238 l    d  .interp  0000000000000000              .interp
0000000000400254 l    d  .note.ABI-tag
0000000000000000 l    df *ABS*  0000000000000000              main2.c
0000000000601038 g        *ABS*0000000000000000              _edata
000000000060103c g     O .bss   0000000000000008              z
0000000000601030 g     O .data  0000000000000008              x
0000000000000000       F *UND*     0000000000000000              addvec
0000000000601018 g        .data  0000000000000000              __data_start
00000000004007e0 g     O .rodata 0000000000000004             _IO_stdin_used
0000000000601028 g     O .data  0000000000000008              y
00000000004006f0 g     F .text  0000000000000047              main

00000000004005c0 <addvec@plt>:
  4005c0:   ff 25 42 0a 20 00      jmpq   *0x200a42(%rip)          # 601008
                                          <_GLOBAL_OFFSET_TABLE_+0x20>
  4005c6:   68 01 00 00 00         pushq  $0x1
  4005cb:   e9 d0 ff ff ff         jmpq   4005a0 <_init+0x18>

00000000004005d0 <__printf_chk@plt>:
  4005d0:   ff 25 3a 0a 20 00      jmpq   *0x200a3a(%rip)          # 601010
                                       <_GLOBAL_OFFSET_TABLE_+0x28>
  ....

00000000004006f0 <main>:
  4006f0:       48 83 ec 08            sub    $0x8,%rsp
  4006f4:       b9 02 00 00 00         mov    $0x2,%ecx
  4006f9:       ba ①_ _ _ _            mov    ■■■■,%edx
  4006fe:       be ②_ _ _ _            mov    ■■■■,%esi
  400703:   bf ③_ _ _ _          mov    ■■■■,%edi
  400708:   e8 ④_ _ _ _          callq  4005c0 <addvec@plt>
  40070d:   8b 0d ⑤_ _ _ _       mov    ■■■■(%rip),%ecx        # 601040 <z+0x4>
  400713:   8b 15 ⑥_ _ _ _       mov    ■■■■(%rip),%edx          # 60103c <z>
  400719:   be e4 07 40 00       mov    $0x4007e4,%esi
  40071e:   bf 01 00 00 00       mov    $0x1,%edi
  400723:   b8 00 00 00 00       mov    $0x0,%eax
  400728:   e8 ⑦ _ _ _ _         callq  4005d0 <__printf_chk@plt>
  40072d:   b8 00 00 00 00       mov    $0x0,%eax
  400732:   48 83 c4 08          add    $0x8,%rsp
  400736:   c3                   retq
```

1）请指出 addvec.c main2.c 中哪些是全局符号？哪些是强符号？哪些是弱符号？以及这些符号经链接后在哪个节？（5 分）

2) 根据上述信息，思考 main 函数中空格①--⑦所在语句所引用符号的重定位结果是什么？请以 16 进制 4 字节数值填写这些空格，将机器指令补充完整（写出任意 3 个即可）。（3 分）

五、综合设计题（共 20 分）

29. 请分别写出 Y86-64 CPU 顺序结构设计与实现中，call XXX 指令在各阶段的微操作(部分微操作已经给出)。另外请给出 Y86-64 流水线实现中，CPU 的冒险的种类和对应处理方法。（10 分）

指令	call XXX
取指	icode:ifun <- M_1[PC]
译码	
执行	
访存	
写回	
更新 PC	

30. 列举几种程序优化的方法，并简述其原理。（至少 2 种）

程序优化: 矩阵 c[n][n] = a[n][n] * b[n][n] ，请针对该程序进行优化，写出优化后的程序，并说明优化的依据。（6 分）

```
int i, j, k;
for(i = 0; i < n; i++)
{
    for(j = 0; j < n; j++)
    {
        C[i][j] = 0;
        for(k = 0; k < n; k++)
            C[i][j] += A[i][k] * B[k][j];
    }
}
```