北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间：2022 年 10 月 31 日 **小班号：**_____

题号	一	二	三	四	五	**总分**
分数						
阅卷人						

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 14 页。

装订线内

不要答题

得分

第一题 单项选择题（每小题 2 分，共 40 分）

注：选择题的回答必须填写在下表中，写在题目后面或其他地方，视为无效。

题号	1	2	3	4	5	6	7	8	9	10
回答	D	D	D	B	C	D	B	B	C	B
题号	11	12	13	14	15	16	17	18	19	20
回答	B	D	B	C	C	D	A	B	A	D

1.已知 a 是 int 类型变量，表达式 a < 0 为真，下列选项中一定为真的表达式是：

A. -a > 0

B. ~a > 0

C. (a / 2) < 0

D. (a >> 1) < 0

答案：D

本题考察整数的运算律，A 选项 a = TMIN 时，-a = TMIN 不满足；B,C 选项中若 a = -1，由补码表示和向零舍入，a/2 = ~a = 0。

2.对于 float 类型变量 a, b, c 下列说法正确的是：

A. 若 a > b，则 a + c > b + c

B. 若 a == b，则 a + c == b + c

C. 若 a + b + c == 0.0，则 c + b + a == 0.0

D. 若 a + b == 0.0，则 b + a == 0.0

答案：D

本题考察浮点数运算律，A 选项 c = INF，B 选项 c = NAN 时不对，C、D 选项变相考察浮点数不满足结合律但满足交换律

3.假设在 C 语言中有如下声明：

```
int i = 0x80000000;
unsigned u = 0x80000001;
short s = 0x8001;
```

则下列表达式结果为真的是：

A. i > (int)u
B. i - 1 > u
C. (unsigned)-1 < i
D. s > u
答案：D
本题考查整型的位级表示，A 选项(int)u 比 i 大 1 所以错误；B 选项左边先溢出为 0x7fffffff 再隐式转换为 unsigned，比右边的 u 小所以也错误；C 选项右边隐式转换成 unsigned 为 0x80000000U，左边为 0xffffffffU，因此左边大于右边，错误；D 选项 s 隐式转化成 unsigned 为 0xffff8001U，明显大于 u，所以正确

4.下列说法正确的是：
A. 在 64 位机器上有 int 型变量 a 和 char 型变量 b，则 sizeof(b) - sizeof(a) < 0 为真
B. 若 x 为整型，则(x >> 1) << 1 <= x 为真
C. 假设 a，b 为 32 位浮点数，ia，ib 是分别与 a，b 位级表示相同的 32 位有符号整型数，则 a < b 等价于 ia < ib
D. 双精度浮点数所能表示的规格化数的阶码范围为 [-127, 126]
答案：B
本题考查整形和浮点数的位级表示，A 选项 sizeof 返回无符号数所以错误，B 选项左边相当于把 x 末位置 0，显然小于等于 x；C 选项考察的是 IEEE 浮点数的比较可以按照位级表示判断，但注意负数需要反过来，所以错误；D 选项应该为[-126, 127].

5.字符串 char s[8]在某小端法存储的 x86-64 机器内存里从低地址到高地址表示为$[32\ 30\ 32\ 33\ 33\ 32\ 30\ 34]_{16}$（不考虑字符串末尾的'\0'），同时在 C 语言中声明 short sh1 = *(short *)(s + 2)和 short sh2 = *(short *)(s + 6)。已知 0 到 9 的 ascii 码分别为 0x30 到 0x39．则下列说法正确的是
A. sh2 - sh1 < 0 为真
B. 语句 printf("%s", s)的输出为 40233202
C. 执行语句 *(int *)(s + 2) = 0x3531 后 printf("%s", s)的输出为 20150004
D. 执行语句 *(int *)(s + 2) = 0x3531 后 printf("%s", s)的输出为 40150002
答案：C 本题考查大小端。sh1 == 0x3332，sh2 = 0x3430，因此 sh2 - sh1 > 0，因此 A 选项错误。B 选项，字符串顺序存储没有大小端一说，所以 s 的内容就是 20233204；如果执行 *(int *)(s + 2) = 0x3531 会把 s 的第 3，4，5，

6 个字节按照小端法覆盖，成为$[32\ 30\ 31\ 35\ 00\ 00\ 30\ 34]_{16}$，因此 s 的内容变成 20150004，所以选 C
（注：该题答案存在问题）

6.如下汇编语句不会产生错误信息的是（　　）

A. movq %rax, $0x123

B. movl %rax, (%rsp)

C. movb %al, %sl

D. movw %si, 8(%rbp)

答案：D

7.考虑下面的 C 语言代码，我们省略了被计算的表达式：

```
long scale2(long x, long y, long z){
    long t = ____________________________;
    return t;
}
```

用 GCC 编译实际的函数得到如下的汇编代码：

```
scale2:
        leaq    (%rdi, %rdi, 2), %rax
        leaq    (%rax, %rsi, 4), %rax
        leaq    (%rax, %rdx, 8), %rax
        ret
```

则 C 代码中缺失的表达式是（　　）

A. ((x + 2) + y + 4) + z + 8

B. 3*x + 4*y + 8*z

C. ((z + 2) + y + 4) + x + 8

D. 3*z + 4*y + 8*x

答案：B

8.考虑下面的 C 语言代码：

```
int comp(data_t a, data_t b){
    return a COMP b;
}
```

代码给出了数据 a 和 b 之间比较的一般形式，这里参数的数据类型 data_t 为某种有符号或无符号的整数类型。COMP 操作通过#define 进行定义。假设 a 在%rdi 中某个部分，b 在%rsi 某个部分。对于下面每个指令序列，组合和描述正确的是（　　）

	指令序列	data_t	COMP
A	cmpl %esi,%edi setl %al	long	<
	指令序列	data_t	COMP
C	cmpb %sil,%dil setbe %al	char	≤

	指令序列	data_t	COMP
B	cmpw %si,%di setge %al	short	≥
	指令序列	data_t	COMP
D	cmpq %rsi,%rdi setne %al	int	!=

答案：B，ACD 分别应该是 int、unsigned char 和 long

9.下列结构体的总大小字节数为 A，重新排列优化后的最小字节数是 B，则 A-B=（ ）

```
struct {
        char *a;   short b;   double c; short d;
        float e;   char  f;   long    g; int    h;
}rec;
```

A. 12　　B. 15

C. 16　　D. 19

答案：C（56-40）。有同学可能忽略 padding，从而算成 52-37=15

10.下面列举了若干技术改进方案，其中不能防止缓冲区溢出攻击的是（ ）

A. 使用金丝雀值（canary）支持对栈破坏检测。

B. 支持变长栈帧。

C. 限制可执行代码区域。

D. 地址空间布局随机化（ASLR）。

答案：B

11.下列 4 组对于 ISA 的描述，它们分别符合(a)_____，(b)_____，(c)_____，(d)_____，的特点.

(a) 某 ISA 中，所有指令均不采用条件码

(b) 某 ISA 中，只有基址和偏移量寻址

(c) 某 ISA 中，执行一些指令需要经过复杂的译码电路

(d) 某 ISA 中，指令长度最短 1 字节，最长 15 字节以上

A. RISC, CISC, CISC, RISC

B. RISC, RISC, CISC, CISC

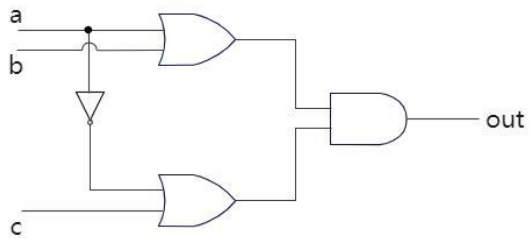
C. CISC, RISC, RISC, CISC

D. CISC, CISC, RISC, RISC

答案：B

不采用条件码，符合 RISC 指令的特征. 只有基址和偏移量寻址，寻址模式单一，也符合 RISC 的描述. 具有复杂指令且需要经过复杂译码电路，符合 CISC 的特征. 指令长度可变并且变化范围达到最少 1 字节最长 15 字节，只能是 CISC

12.对应下述组合电路的正确 HCL 表达式为：



A. Bool out = (a && b) || (!a && c)
B. Bool out = (!a && b) || (a && c)
C. Bool out = (a || b) && (a || c)
D. Bool out = (a || b) && (!a || c)

答案：D；**本体主要考察组合逻辑电路和 HCL 表达式。逻辑电路由 2 个或门、1 个与门、1 个反向器构成，难度较易。**

13.将一个延迟为 300ps 的组合逻辑划分为三级流水线，时钟寄存器的延迟为 20ps，则该流水线的吞吐量为：
A.10GIPS
B.8.33GIPS
C.3.33GIPS
D.2.77GIPS

答案：B；1 / (100ps + 20ps) = 8.33GIPS

14.如果在 SEQ 中添加一条新的指令 ircmovq，其作用是根据当前处理器的条件码来判断是否需要将一个立即数传送到寄存器内，则下列叙述错误的是：
A. 需要增加一个新的 icode 标识符
B. 该指令的长度为 10 字节
C. 只需要修改 SEQ 处理器的执行阶段的硬件逻辑
D. 在执行阶段 valE 信号的值会被赋值为 valC 的值

答案：C，可以和 rrmovq 指令共用一个 icode

15．下面说法正确的是（　　）。

A． 随着循环展开次数越多，分支预测次数更少，指令调度空间更大，代码性能更好。

B． 编译优化无法跨越函数调用和内存别名的阻碍。

C． 算法的渐进复杂度对于程序优化十分重要。

D． 由于处理器硬件带来的延迟界限（latency bound）是程序性能的终极限制。

答案：C

A 循环展开次数增加会导致代码膨胀、增大寄存器压力，影响性能（课本 5.11.1）。

B 编译器能处理函数调用和内存别名，只是代价很高；（课本章节 5.1 用词“限制了可能的优化”“大多数编译器不会”。此外，函数内联优化就是一个很好的反例）。

C 很直白的正确的话，用来在这里坑一下喜欢猜答案的同学。

D 吞吐量界限才是程序性能的终极限制（课本章节 5.6 最后一句话）。

16．以下关于存储器的说法中，错误的是：

A．对于旋转磁盘（rotating disk），可以通过提高旋转速率（rotational speed）的方式减少旋转时间（rotational latency）和传送时间（transfer time），从而降低访问时间（access time）。

B．在 CPU 向磁盘控制器（disk controller）发送读取数据请求后，磁盘控制器会读取指定扇区的内容，并将其传送到内存的指定位置，这一传送过程不需要 CPU 参与，称为 DMA（direct memory access）传送。

C．对于同一个 SSD，读取速度通常比写入速度快。

D．SRAM 是非易失性（nonvolatile）存储器，其数据在断电后不会丢失。

答案：D。SRAM 和 DRAM 都是易失性存储。

难度：简单

17．以下关于存储器的说法中，错误的是：

A． SDRAM 同时具备 SRAM 和 DRAM 的特点。

B．DRAM 和磁盘的性能发展滞后于 CPU，现代处理器为了弥补 CPU 的访存需求和内存延迟的差距，频繁地使用高速缓存。

C． SRAM 单元比 DRAM 单元使用更多的晶体管，因而密集度低，而且更贵。

D．电路设计者将 DRAM 组织成二维阵列而不是线性数组的一个原因是降低芯片上地址引脚的数量。

答案：A。SDRAM 的“S”是指 Synchronous。SRAM 的“S”是指 Static。

难度：中等。

18．如果我们希望将原来 4MB 的 cache 调整为 6MB，可以采取的做法是（　　）。

A. 将 S 从 4096 调整为 6144。
B. 将 E 从 16 调整为 24。
C. 将 B 从 64 调整为 96。
D. 以上答案都不对。
答案：B
此题源于 Intel Core2 Duo T9300 的设计。S 和 B 需要是 2 的 n 次方，但 E 不需要。

19. 如果一个系统中访问下一级存储器的传输时间远大于 cache 的处理速度，我们尝试引入 cache 来改善一个局部性良好的程序的 IO 时，应当使用的写策略是
① 写回(write-back) ②直写(write-through)
③ 写分配(write-allocate) ④非写分配(no-write-allocate)
A. ① ③
B. ② ③
C. ① ④
D. ② ④
答案：A
难度：易
题目描述希望尽量减少总线流量，因此我们选择写回（一般搭配写分配），这里特意强调了局部性良好的程序，是为了消除写分配的 disadvantage that every miss results in a block transfer from the next lower level to the cache，因此这里选择写回+写分配的理由充分。大约需要 0.5～1 分钟。

20. 执行以下函数，消耗的内存带宽约为（）。
已知：

1. Write miss 发生时采用写分配（write-allocate）
2. X、Y、result 三个数组在函数执行前均不存在于 CPU cache 中，函数执行完毕后均写入内存中
3. 不考虑 SIMD 指令，不考虑编译优化（包括循环展开等）
4. Cache line size 为 64 Bytes，内存到 cache 的传输以 cache line 为单位

```
void saxpy(int N,
    float scale,
    const float X[],
    const float Y[],
    float result[]) {
    for (int i = 0; i < N; i++) {
        result[i] = scale * X[i] + Y[i];
```

```
    }
}
```

（提示：1．只需考虑对 X、Y、result 三个数组的读写；2．消耗的内存带宽=从内存中读取的字节数+写入内存的字节数）

A．3N Bytes　B．4N Bytes　C．12N Bytes　D．16N Bytes

答案：D

难度：难

解析：本题容易误选为 C。因为写分配，CPU 会把 result 数组也读入 cache 中。具体而言，假定 result 数组的起始地址 64 字节对齐，那么写入 result[0]的时候，cache 里会为 result[0..15]分配空间，因而 result[1..15]都需要读入 cache；再考虑内存到 cache 传输的最小单位是 cache line，那么 result[0]也要读入

背景：本题是性能分析中的常见误区。内存带宽计算是性能瓶颈分析的重要一环（在并行计算中尤其如此），忘记考虑 write-allocate 是一大失误。

得分

第二题 类似 IEEE 浮点数标准定义 8 位浮点数，符号位、阶码位和尾数位长度分别为 1,3,4，依此回答下列问题：

1. 若浮点十六进制表示为 0xBD，那么该数的十进制大小为_____；若十进制大小为 9/64，那么浮点十六进制表示为_____
2. 已知浮点数 A 和 B，A 浮点表示为 010000xx，B 的浮点表示为 00111x11，x 表示对应的位未知，则 A_____B（请填小于、大于、等于或无法判断）
3. 若某个 unsigned char 类型数的十六进制表示为 0x6E，那么该数的浮点表示为_____
4. 该标准浮点数能够精确表示的最大整数的二进制表示是_____
5. 用上述浮点格式表达 unsigned char 类型整数，是否会发生溢出____（请填是或否），是否会发生舍入_____（请填是或否）
6. 如果有另一种符合 IEEE 标准的浮点数，符号位、阶码位和尾数位长度分别为 1，4，3，则其能精确表示的实数的数量____（请填更多、更少或相同）
7. 再考虑另一个符合 IEEE 标准的 16 位浮点系统，包含 1 位符号，m 位阶码，n 位尾数．已知实数 1 在该浮点格式系统中的位级表示为 0x3c00，则 m = _____，n = _____．该浮点类型所表示的最小负非规格化数（十进制）为 _____．

答案：

1) -1.8125 0x09　　二进制浮点数到十进制浮点数的转化运算
2) 大于　　浮点数的大小比较，先看符号位，再看阶码位
3) 01110000　　整数到浮点数的转化运算，此处为上溢出的情况
4) 1111_2　　考察浮点格式
5) 是 否　　阶码位最大为 $110_2=6_{10}$，6-3=3，能精确表达的值为 1111_2，故会发生溢出，但不会发生舍入
6) 更多　　浮点数格式的考察
7) 5, 10, -1023/(2^24)　在符合 IEEE 标准的浮点系统下，实数 1 的表示为阶码为 0111...1，其余位为 0．所以 m = 5，n = 10．此浮点系统能表示的最小负非规格化数的位级表示为 1000 0011 1111 1111，即 -1023/(2^24)．

第 1-6 题每题 2 分（两空的每空 1 分，一空的 2 分），第 7 题 3 分（每空 1 分）

得分

第三题 下列的 C 代码描述了一个 switcher：

```
struct prob{
  long *p;
  struct { long x; long y; long z;}s;
  struct prob *next;
};
void switcher(long a, long b, long c, struct prob *sp){
  long val;
  switch(a){
    case ①_____: c = ⑧_____;
    case ②_____: val = ⑨_____; break;
    case ③_____: c = ⑩_____;
    case ④_____: val = ⑪_____; break;
    case ⑤_____: val = ⑫_____; break;
    case ⑥_____: ⑬_____;break;
    case ⑦_____: val = ⑭_____;break;
    default:⑮_____; break;
  }
  sp->s.y = val;
}
```

采用 GCC 编译器产生的汇编代码如下所示，请根据此进行分析，补全上述代码。

```
switcher:
        cmpq    $7, %rdi
        ja      .L2
        jmp     *.L4(,%rdi,8)
.L4:
        .quad   .L3
        .quad   .L5
        .quad   .L6
        .quad   .L7
        .quad   .L2
        .quad   .L8
        .quad   .L9
        .quad   .L10
```

```
.L8:
        movq    %rsi, %rdx
        xorq    $31, %rdx
.L6:
        leaq    2022(%rdx), %rax
        movq    %rax, 16(%rcx)
        ret
.L3:
        subq    $54, %rdx
.L5:
        addq    %rdx, %rsi
        leaq    (%rsi,%rsi,2), %rax
        movq    %rax, 16(%rcx)
        ret
.L7:
        movq    8(%rcx), %rax
        movq    %rax, 16(%rcx)
        ret
.L10:
        movq    $1898, 24(%rcx)
        movq    %rax, 16(%rcx)
        ret
.L9:
        movq    (%rcx), %rax
        movq    (%rax), %rax
        movq    %rax, 16(%rcx)
        ret
.L2:
        movq    %rcx, 32(%rcx)
        movq    %rax, 16(%rcx)
        ret
```

参考答案如下，此题源于习题 3.31 和 3.41 的合并，因而可以算作简单题。

```
void switcher(long a, long b, long c, struct prob *sp){
  long val;
  switch(a){
```

case 5 : c = b^31 ; case 2 : val = c+2022 ; break;
case 0 : c = c-54 ; case 1 : val = (c+b)*3 ;break;
case 3 : val = sp->s.x ;break;
case 7 : sp->s.z = 1898 ;break;
case 6 : val = *(sp->p) ;break;
default: sp->next = sp ;break;

```
    }
    sp->s.y = val;
}
```

补充说明：
上面的每个单元格之间原则上是可以互换的，但应保持程序基本语义不变。

(c+b)*3 写成(b+c)*3 或者 3*(b+c)或者 3*(c+b)也都算对。有同学写成了 3*b+3*c，周一阅卷的时候这样的写法没有给分，但后来我们注意到有的编译器可能会将其优化为题目中的汇编代码。所以这样的书写也可以给分。

凡是不符合 c 语言语法的写法都不能给分。例如有同学写成 3(b+c)的不能得分，也有同学写成了 b xor 31 的同样不能给分。

得分

第四题 请分析 Y86-64 ISA 中新加入的一组条件 POP 指令：cpopqXX，cpopqXX 和 popq 指令的机器码格式如下。

cpopqXX	B	fun	rA	F
popq	B	0	rA	F

类似cmovXX，该组cpopqXX指令在条件码(Cnd)满足时，会将栈顶的8字节数据读入到寄存器rA中，并且使栈指针增加8；如果条件不满足，则不执行该指令，即对rA、栈指针都无影响。

1.若在教材所描述的SEQ处理器上实现该指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd；the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

Stage	cpopqXX rA
Fetch	icode:ifun ← M_1[PC] rA:rB ← M1[PC+1] valP ← PC+2
Decode	
Execute	
Memory	
Write back	
PC update	PC ← valP

2.考虑在教材中的 Pipeline 处理器上实现该指令，需要改进教材所描述的 PIPE 处理器在执行（E:Execute）阶段的处理逻辑，使其能够在条件不满足时不修改 rA 和栈指针的值。请依据改进后的处理器，补全执行阶段的部分 HCL 代码：

```
word e_dstE = [
    (E_icode in {IRRMOVQ, ①_____ } && ②_____) : RNONE;
    1 : E_dstE;
];

word e_dstM = [
    (E_icode == ①_____ && ②_____) : RNONE;
    1 :  E_dstM;
]
```

3.和 popq 与 mrmovq 相同，在执行该指令的过程中有可能会引发加载-使用冒险（load-use hazard）。请将在引入了 cpopqXX 指令后的 Y86-64 PIPE 处理器中，加载-使用冒险的触发条件以及处理冒险的控制逻辑补充完整。如有多种可能，任写一种即可：

触发条件：E_icode in {IMRMOVQ, IPOPQ} && e_dstM in {____, ____}

控制逻辑：（选填 stall, bubble 和 normal）

F	D	E	M	W
			normal	normal

4. 基于改造后的 Y86-64 PIPE 考虑如下代码片段，回答问题。

```
# long foo(long n)
# n in %rdi
foo:
    xor     %rax, %rax                  # Line 1
    pushq   %rsp                        # Line 2
    irmovq  $1, %rsi                    # Line 3
L1:
    subq    %rsi, %rdi                  # Line 4
    andq    %rdi, %rdi                  # Line 5
    cpopqe  %rsi                        # Line 6
    addq    %rsi, %rax                  # Line 7
    andq    %rdi, %rdi                  # Line 8
```

```
    jne     L1                              # Line 9
    ret                                     # Line 10
```

假设条件分支总是预测跳转，计算在输入的n值取1和3时，函数foo执行的周期数。（周期数计算从执行foo第一条指令开始，直到其返回指令ret完全通过流水线为止。另外在调用foo之前，处理器已经执行过足够多的nop指令。）

n为1时候需要执行_____个周期；n为3时候需要执行_____个周期。

答案：

1.

Stage	cpopqXX rA
Fetch	icode:ifun ← M_1[PC] rA:rB ← M1[PC+1] valP ← PC+2
Decode	valB ← R[%rsp] valA ← R[%rsp]
Execute	valE ← valB + 8 Cnd ← Cond(CC, ifun)
Memory	valM ← M_8[valA]
Write back	if (Cnd) R[%rsp] ← valE if (Cnd) R[rA] ← valM
PC update	PC ← valP

（每空1分，共7分）

2. ①IPOPQ ②!e_Cnd （每空1分，共2分），第1空填ICPOPQXX也算对

3.

触发条件：E_icode in {IMRMOVQ, IPOPQ} && e_dstM in {d_srcA, d_srcB}

控制逻辑：（选填 stall, bubble 和 normal）

F	D	E	M	W
Stall	Stall	Bubble	normal	normal

（共 3 分，其中 e_dstM in {d_srcA, d_srcB} 1 分，全对才得分；控制逻

辑 2 分，全对才得分）
4.

解析：循环过程为Line 4-9,共6条
n=3 ： 4(填充流水线)+3(Line 1~3)+6*3(共三次循环)+1(Line 7-8:load-use hazard)+2(Line 9:branch misprediction)+1(ret指令)=29
n=1 ： 4(填充流水线)+3(Line 1~3)+6 (第一次循环)+1(Line 7-8:load-use hazard)+2(Line 9:branch misprediction)+1(ret指令)=17
(对任意一空得2分，两空全对得3分)

得分

第五题 请利用本课程中的“高速缓存”的相关知识，回答下列问题。

1.在一个 **8-bit** 地址空间的机器上，考虑如下的高速缓存：此高速缓存共有 4 组 **(S=4)**，为二路组相联**(E=2)**，Tag 段的大小为 3-bit**(t=3)**，替换策略为 **LRU**。初始状态下各组的有效位（V）和 Tag 位如下表所示：

	V	Tag	V	Tag
SET 0	1	000	0	001
SET 1	0	001	1	110
SET 2	0	110	1	101
SET 3	1	010	0	001

（1）此 cache 所能存储的数据总容量为 C = _____ Bytes．（3 分）

（2）现要读取地址空间 0xd2 处的数据，此访存是否命中 cache？_____（填“是”或“否”）。（2 分）请在下表中写出经过此次访存后，更新后的 cache 的有效位（V）和 Tag 位。（仅写出有修改的部分即可，如果你认为没有修改，此表可以为空）（2 分）

	V	Tag	V	Tag
SET 0				
SET 1				
SET 2				
SET 3				

（3）在（2）进行访存的基础上，考虑如下的访存序列：0xc8, 0x16, 0x01, 0xb5（均为读取一个字节的数据）。这四次数据访问的**命中率 (hit rate)** 为_____。（2 分）假设此 cache 的**命中时间 (hit time)** 为 10 个周期，**不命中处罚 (miss penalty)** 为 190 个周期，则这四次数据访问的总延迟为_____周期。（2 分）

2.下面我们将考虑实际程序中发生的访存行为。考虑一个 32-bit 地址空间的机器，并有如下的高速缓存：总大小为 16 Bytes，每个 cache block 的大小为 4 Bytes，组织结构为全相联（S=1），替换策略为 LRU。

在此机器上运行如下 C 语言代码。假设初始时 cache 为空，只考虑读写数据引起的 cache 访问，不考虑编译优化，临时变量 a, b, c, i 均储存在寄存器中，int 数据类型的大小为 4 Bytes，f 数组的起始地址为 0x10000000。回答下列问题：

```
1: # define N 20
```

```
2: int f[N + 2];
3: f[0] = 1;
4: f[1] = 1;
5: for(int i = 0; i < N; i++) {
6:    int a = f[i];
7:    int b = f[i + 1];
8:    int c = a + b;
9:    f[i + 2] = c;
10: }
```

（1）假设 cache 在写命中时采用直写(write-through) 策略，在写不命中时采用非写分配(not-write-allocate) 策略，则当程序第一次运行到第 6 行时，所发生的访存是否命中 cache？_____（填“是”或“否”）。（2 分）整个程序的执行过程共命中 cache _____次。（1 分）

（2）假设 cache 在写命中时采用写回(write-back) 策略，在写不命中时采用写分配(write-allocate) 策略，整个程序的执行过程共命中 cache _____次。（1 分）

1.（1）**.64**（3 分）

由 S=4 可得 s=2，故 b=8-s-t=3，即每个 cache block 的大小为 B=23=8 Bytes。Cache 总容量 C=S*E*B=64 Bytes.

本题考察 cache 的基本概念，属于容易题。

（2）**.否**（2 分）

	V	Tag	V	Tag
SET 0				
SET 1				
SET 2	1	110		
SET 3				

0xd2 的二进制表示为 110 10 010，可得其组号为 set 2。在当前状态下 set 2 虽然有 tag 为 110 的 cache block，但此 block 的有效位 V=0，所以不命中。由于 set 2 有一个 V=0 的 cache block，所以 cache 更新后，应当将这个 block 的有效位设置为 1，并将其 tag 位设置为 110.

本题考察访问 cache 的基本操作，属于容易题。

注意：填写此表时只要表达的意思对即可给分。（比如只写了 set 2 的第一个 block 的有效位，或者将其他没有改动的地方抄了下来都应算对）

（3）**.50%**（2 分）；**420**（2 分）

0xc8 的二进制为 110 01 000，命中 set 1。0x16 的二进制为 000 10 110，查找 set 2，由于没有标签为 000 的 block，所以不命中。根据 LRU 策略，set 2 中 tag 为 101 的 block 被替换出去。0x01 的二进制为 000 00 001，查找 set 0，其中 tag 为 000 的 block 有效位为 1，命中。0xb5 的二进制为 101 10 101，查找 set 2，由于 tag 为 101 的 block 已经被替换出去了，所以不命中。
综上，命中率为 2/4*100%=50%.
总延迟为 2*10+2*(10+190)=420 周期。
此题考察 cache 的访存行为，较为综合，时间可能花费较长。

2.（1）**.否**（2 分）；**19**（1 分）
由于 cache 采用非写分配策略，所以代码运行到 3、4 行时并没有将 f[0]和 f[1]读入 cache，因此在第一次执行到第六行读取 f[0]时是不命中的。
第一次循环体执行时，所有访存均不命中。当第 i（i>1）次循环执行时，第 6 行的读数据命中 cache，第 7 行的读数据的地址由于在第(i-1) 次循环是写不命中，而 cache 采取非写分配策略，因此该 block 还没有被加载到 cache 中，所以仍是不命中。第 9 行的写是第一次访问该地址，因此也是不命中的。即在第二次即以后每次执行循环体时，会有一次读命中、一次读不命中和一次写不命中。循环体共执行 20 次，所以共命中 cache 19 次。
本题考查对 cache 写策略的理解。计算命中次数需要发现一定规律，难度中等。
（2）**.40**（1 分）
由于本题中 cache 采用写分配策略，因此第一次写不命中时就会把相应的 block 加载到 cache 中，从而在下一次读的时候就可以命中了。在每次循环体执行时，第 6 行和第 7 行的读数据是命中的，第 9 行的写数据是不命中的。循环体共执行 20 次，故命中 cache 总次数为 4.
本题同上题，考察对 cache 写策略的理解。同样需要发现一定规律，难度中等。