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：2018 年 11 月 14 日**小班号：**___**小班教师：**_____

题号	一	二	三	四	五	六	**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　　页。

得分

第一题 单项选择题（每小题 2 分，共 30 分）

注：选择题的回答请填写在下表中。

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

1. 下列哪种类型转换既可能导致溢出、又可能导致舍入？

A) int 转 float B) float 转 int C) int 转 double D) float 转 double

答案：b，本题考查浮点数类型转换，参见中文版 86 页。

2. 在采用小端法存储机器上运行下面的代码，输出的结果将会是？

（int,unsigned 为 32 位长,short 为 16 位长,0~9 的 ASCII 码分别是 0x30~0x39）

```
char *s = "2018";
int *p1 = (int *)s;
short s1 = (*p1)>>12;
unsigned u1 = (unsigned) s1;
printf("0x%x\n",u1);
```

A) 0x00002303 B) 0x00032303 C) 0xffff8313 D) 0x00008313

答案：C

本题考查大小端存储，以及整数类型转化。字符串在存储时不区分大小端，前面的字符存储在低地址，而在被转化成整型的时候低地址被视为低位，因此*p1 为 0x38313032，s1 为 0x8313。在由 short 转化为 unsigned 的时候，我们要先改变大小，之后完成有符号到无符号的转换，因此 u1 为 0xffff8313。

3. 考虑如下函数

```
void XOR(int x, int y) {
        y = x ^ y;
        x = x ^ y;
        y = x ^ y;
        printf(x, y);
```

```
}
```

XOR(a,b)的输出结果为？

A) a, b　　B) b, a　　C) b,0　　D) b, a^b

答案：C
本题考查 xor 的特殊用法，x xor x = 0 ， 0 xor x = x ；
答案修订：应为 B，该代码将 x, y 值进行交换。另外，题面应为 printf(“%d, %d”, x, y)；

4. 在 x86-64 下，以下哪个选项的说法是错误的？
A) movl 指令以寄存器作为目的时，会将该寄存器的高位 4 字节设置为 0
B) cltq 指令的作用是将%eax 符号扩展到%rax
C) movabsq 指令只能以寄存器作为目的
D) movswq 指令的作用是将零扩展的字传送到四字节目的

答案：D
movswq 应该是符号扩展

5. 以下代码的输出结果是

```
union {
    double d;
    struct {
        int i;
        char c[4];
    } s;
} u;
u.d = 1;
printf("%d\n", u.s.c[2]);
```

A) 0　　B) -16　　C) 240　　D) 191

答案：B
c[2]保存的数据是$(11110000)_2$，char 保存的数据是有符号数，所以应该是-16
答案修订：ABC 均正确。

题目未说明大小端，若为大端法，选择 A。若为小端法，u.s.c[2]为$(11110000)_2$，由于 char 在不同的编译器上会被实现成 signed char 或 unsigned char，因此扩展为 int 时-16 和 240 均有可能，B，C 均正确。

6. 下列关于 C 语言中的结构体(struct)以及联合(union)的说法中，正确的是：

A) 对于任意 struct，将其成员按照其实际占用内存大小从小到大的顺序进行排列不一定会使之内存占用最小

B) 对于任意 struct，将其成员按照其实际占用内存大小从小到大的顺序进行排列一定不会使之内存占用最大

C) 对于任意 union，将其成员按照其实际占用内存大小从小到大的顺序进行排列不一定会使之内存占用最小

D) 对于任意 union，将其成员按照其实际占用内存大小从小到大的顺序进行排列一定不会使之内存占用最大

答案：A。联合以及只含一个基本数据类型成员的结构体的内存占用与其成员排列方式无关，即任意排列方式都可使得内存占用最小（最大），故 b、c、d 错误。对于 struct，应当将成员按照其存储单元所占内存大小从小到大（或从大到小）的顺序进行排列才能使内存占用最小，故 a 正确。

7. 下列关于程序控制结构的机器代码实现的说法中，正确的是：

A) 使用条件跳转（conditional jump）语句实现的程序片段比使用条件赋值（conditional move）语句实现的同一程序片段的运行效率高

B) 使用条件跳转语句实现的程序片段与使用条件赋值语句实现的同一程序片段虽然效率可能不同，但在 C 语言的层面上看总是有着相同的行为

C) 一些 switch 语句不会被 gcc 用跳转表的方式实现

D) 以上说法都不正确

答案：C。条件跳转本身开销大于条件赋值，但条件赋值会将两个分支中的运算都完成，故分支中的运算较为复杂时，使用条件赋值语句实现的程序效率较低，故 a 错误。分支中的运算带有副作用时，条件跳转语句和条件赋值语句实现的程序行为不同，故 b 错误。switch 语句中的 case 若比较稀疏，则不会被用跳转表的方式实现，故 c 正确。

8. 下列关于条件码的描述中，不正确的是（）

A) 所有算术指令都会改变条件码

B) 所有比较指令都会改变条件码

C) 所有与数据传送有关的指令都会改变条件码

D) 条件码一般不会直接读取，但可以直接修改

答案：C，leaq 是 movq 指令的变形，它只传送地址，不改变条件码。
答案修订：ABCD 均给分。
AB 没有指明是 x86 指令系统；x86 指令系统中有 SIMD 类指令不改变条件码；
C 数据传送一般不改变条件码；
D 正确，SAHF、STC、CLC、CMC 等指令均可以直接写条件码。

9. 请比较 RISC 和 CISC 的特点，回答下述问题：
假设编译技术处于发展初期，程序员更愿意使用汇编语言编程来解决实际问题，那么程序员会更倾向于选用 _____ ISA。
假设你设计的处理器速度非常快，但存储系统设计使得取指令的速度非常慢（也许只是处理单元的十分之一）。这时你会更倾向于选用 _____ ISA。
A) RISC、RISC B) CISC、CISC C) RISC、CISC D) CISC、RISC
答案：B
//知识点 1：CISC 有更多的指令，有些更接近高级语言
//知识点 2：CISC 的指令功能更复杂，指令执行需要更多的周期，一定程度上可以平衡处理速度与指令访存的速度差异。不过，通常处理器设计中，主要通过多层次的存储体系结构来弥补两者之的速度差异。

10. Y86 指令 **popl rA** 的 SEQ 实现如下图所示，其中❶和❷分别为：

Fetch	icode:ifun ← M_1[PC] ra:rb ← M_1 [PC+1] valP ← ❶
Decode	valA ← R[%esp] valB ← R[%esp]
Execute	valE ← ❷
Memory	valM ← M_4 [valA]
Write Back	R[%esp] ← valE R[ra] ← valM
PC Update	PC ← valP

A) PC + 4 valA + 4
B) PC + 4 valA + (- 4)
C) PC + 2 valB + 4
D) PC + 2 valB + (- 4)

答案：C
//知识点：popl 弹栈指令，栈结构

1） popl 是双字节指令，下一条指令位置 PC+2

2） Y86 是 32 位体系结构，popl 弹出 4 字节，弹栈栈指针增加，valA + 4

11. 假设已有声明 int i, int sum, int *p, int *q, int *r, const int n = 100, float a[n], float b[n], float c[n], int foo(int), void bar(), 以下哪项程序优化编译器总是可以进行？

A	for(i = 0; i < n; ++i) {
	  a[i] += b[i];
	  a[i] += c[i];
	}
		float tmp;
		for(i = 0; i < n; ++i) {
		  tmp = b[i] + c[i];
		  a[i] += tmp;
		}

B	*p += *q;
	*p += *r;
		int tmp;
		tmp = *q + *r;
		*p += tmp;

C	for(i = 0; i < n; ++i)
	  sum += i * 4;
		int N = n * 4;
		for(i = 0; i < N; i += 4)
		  sum += i;

D	for(i = 0; i < foo(n); ++i)
	  bar();
		int tmp = foo(n);
		for(i = 0; i < tmp; ++i)
		  bar();

答案：C

A 浮点数不满足结合律

B p,q,r 可能指向同一个地址

D foo 函数可能有副作用

12. 设一种缓存共包含 4 个缓存块，每个缓存块 32 字节，采用 LRU 替换算法。设缓存初始状态为空，对其进行下列 char 类型内存地址序列访存，0x5a7, 0x5b7, 0x6a6, 0x5b8, 0x7a5, 0x5b9。对于直接映射缓存和 2 路组相联缓存，分别能产生几次命中？

A) 1, 3

B) 1, 4

C) 2, 3

D) 2, 4

答案： A

考核地址中 TAG 位域/组索引位域/缓存块内偏移位域在直接映射和组相联的定义，LRU 替换算法的概念和过程。

13. 设一种全相联缓存共包含 4 个缓存块，如果循环地顺序访问 5 个不同的缓存块，下列哪种替换算法会产生最多的命中？

A) 最近最少使用替换策略 LRU

B) 先入先出替换策略 FIFO

C) 随机替换策略 Random

D) 后入先出替换策略 LIFO

答案：A

考核替换算法的概念和过程

答案修订：CD 均正确。

A LRU 命中率 0%；

B FIFO 命中率 0%；

C Random 每次新来一个块，随机换出的块会导致之后第 1,2,3,4 次访存有一次 miss（等概率），每次 miss 平均 2.5 次访存，每次访存 miss 率 40%，因此命中率 60%；

D LIFO 命中率 60%，最后两个块交替发生 miss。

14. 以下关于存储的描述中，正确的是（ ）

A) 由于基于 SRAM 的内存性能与 CPU 的性能有很大差距，因此现代计算机使用更快的基于 DRAM 的高速缓存，试图弥补 CPU 和内存间性能的差距。

B) SSD 相对于旋转磁盘而言具有更好的读性能，但是 SSD 写的速度通常比读的速度慢得多，而且 SSD 比旋转磁盘单位容量的价格更贵，此外 SSD 底层基于 EEPROM 的闪存会磨损。

C) 一个有 2 个盘片、10000 个柱面、每条磁道平均有 400 个扇区，每个扇区有 512 个字节的双面磁盘的容量为 8GB。

D) 访问一个磁盘扇区的平均时间主要取决于寻道时间和旋转延迟，因此一个旋转速率为 6000RPM、平均寻道时间为 9ms 的磁盘的平均访问时间大约为 19ms。

答案：B

A 选项中 SRAM 和 DRAM 位置反了。

C 选项中，硬盘容量 1GB=10^9 Byte，因此容量应该为 8.192GB。

D 选项中，平均旋转延迟为 0.5*(60s/6000RPM)=5ms，平均访问时间为 14ms。

15. 某计算机地址空间 12 位，L1 cache 大小为 256 字节，组数 S=4，路数 E=2，现在在地址 0x0 处开始有一个 N 行 M 列的 int 类型的数组 int A[N][M]，有如下 C 代码：

```
int ans=0;
for(int j=0;j<M;++j)
    for(int i=0;i<N;++i)
        ans+=A[i][j];
```

不考虑并行、编译优化等等会影响访问 A 数组元素顺序的因素，则如下(N,M)对中会导致全部 Cache Miss 的有（ ）

(N,M)= (64,4)， (32,8)， (16,16)， (2,128)

A) 4 个

B) 3 个

C) 2 个

D) 1 个

答案：C

每个缓存块可以存放 256/S/E/4=8 个 int 类型的数据

1、(N,M)=(64,4)时，访问数组第一行元素会将第一、二行元素都放入缓存，因此在访问数组第二行元素 A[1][0]时不会 Cache Miss

2、(N,M)=(32,8)时，可以验证全部 Cache Miss

3、(N,M)=(16,16)时，可以验证全部 Cache Miss

4、(N,M)=(2,128)时，访问 A[0][1]时不会 Cache Miss

得分

第二题（10 分）

在本题中，int 和 unsigned 均为 32 位。int 用补码表示，算术右移。float 为 32 位 IEEE754 标准，double 为 64 位 IEEE754 标准。

1. （2 points）生成任意 int 类型的 x，然后将它转换为 unsigned 类型：

```
int x = random();
unsigned ux = (unsigned) x;
```

对于以下每个 C 语言表达式，你需要判断它的值是否恒为 1。如果是，圈出 Y；否则圈出 N。

表达式	恒为 1?
(x >= 0) || (x < ux)	Y N
((x>>1) << 1) <= x	Y N

2. （2 point）请补全函数 nlz，使之可以用来计算 unsigned 类型数据的二进制表示中**前导零**的个数（提示：填写数字）：

```
// TODO: Complete the `nlz` function to
//       count the number of leading zeros
int nlz(unsigned x) {
  double w = 0.5;
  double y = (double)x + w;
  long long z;
  memcpy(&z, &y, sizeof(y));
  return ____ - (z >> ____);
}
```

3. （2 point）上式中 w 的取值并不唯一。请判断以下哪个/哪些数字也符合 w 的要求：

(a) 0.2 (b) 0.3 (c) 0.9 (d) 1.2

4. （4 points）下面所示的 strlen_x 是字符串求长度的函数在**大端序**电脑中的一种实现，nlz 函数的意义如前所述。

```
int func_x(unsigned x) {
  unsigned u = 0x7F7F7F7F;
  unsigned y = (x & u) + u;
  y = ~(y | x | u);
  return nlz(y) >> 3;
}

size_t strlen_x(const char *s) {
  size_t t, n = 0;
  const unsigned *si = (const unsigned *)s;
  while ((t = func_x(*si++)) == 4) { n += t; }
  return n + t;
}
```

不改变 strlen_x 的实现，请修改 func_x 以使得 strlen_x 适用于**小端序**的电脑（提示：仅使用常数、算术/逻辑运算符和已定义变量）：

```
int func_x(unsigned x) {
  unsigned u = 0x7F7F7F7F;
  unsigned y = (x & u) + u;
  y = ~(y | x | u);
  return (___- nlz((___) & (y-1))) >> 3;
}
```

答案：

1.

表达式	恒为 1?
(x >= 0) || (x < ux)	N: e.g. x = -1.
((x>>1) << 1) <= x	Y: x>>1 rounds toward minus infinity.

2. 32 - 1 - ((z >> 52) - 1023)，即应该填 **1054**。

3. w 的取值范围为[0.5, 1)，因此只有 **c** 符合要求。

4. 原始的 func_x 函数的意义在于在 unsigned 类型数据的四个字节中，从 most significant byte 到 least significant byte 寻找第一个等于’\0’的字节的位置（0~3），若找不到则返回 4，因此大端序机器上正好可以用来做 strlen_x 的累加。而在一台小端序机器上，const unsigned 类型的指针会将顺序排列的四个 char 颠倒顺序来解读，因此 func_x 的意义应改变为从 least significant byte 开始寻找第一个等于’\0’的字节的位置。仔细考察 y 的含义，可以构造出以下式子：

```
// TODO: Fill in the blank line
int func_x(unsigned x) {
  unsigned u = 0x7F7F7F7F;
  unsigned y = (x & u) + u;
  y = ~(y | x | u);
  return (32 – nlz(~y & (y-1))) >> 3;
}
```

得分

第三题（15 分，每空 1 分）

数据结构定义如下：（测试用机：64 位 Linux 机器）

```
typedef struct s1 {
         char cc[N];
         int ii[N];
         int *ip;
} S1;

S1 t1[N];
```

1、计算该数据结构的空间：

当 N=3 时，sizeof(S1) = ____(1)____，sizeof(t1) = ____(2)____

当 N=4 时，sizeof(S1) = ____(3)____，sizeof(t1) = ____(4)____

当 N=5 时，sizeof(S1) = ____(5)____，sizeof(t1) = ____(6)____

答案：

（1）24 （2）72

（3）32 （4）128

（5）40 （6）200

2、当 N=4 时，该数据结构初始化代码如下：

```
void init(int n)
{
        int i;
        for (i=0; i<n; i++) {
                t1[i].ip = &(t1[i].ii[i]);
        }
}
```

根据上述代码，填写下面汇编中缺失的内容：

```
init:
        movl    $0, %ecx
```

```
        jmp     .L2
.L3:
                movslq  %ecx, %rax
                leaq    (  (1)  ,%rax,8), %rsi
                leaq    0(,%rsi,4), %rdx
                addq    $t1+4, %rdx
                salq    (2), %rax
                movq    (3), (  (4)  )
                addl    $1, %ecx
        .L2:
                cmpl    (5), %ecx
                jl      .L3
                rep ret
```

答案：

（1）%rax（2）$5 （3）%rdx （4）t1+24(%rax) （5）%edi

答案修订：(4) 全给分，卷面多了一对括号。

3、当 N=3 时，函数 fun 的汇编代码如下：

```
fun:
        movslq  %esi, %rax
        movslq  %edi, %rdi
        leaq    (%rdi,%rdi), %rdx
        leaq    (%rdx,%rdi), %r8
        leaq    (%r8,%r8), %rcx
        addq    %rcx, %rax
        movl    %esi, t1+4(,%rax,4)
        addq    %rdx, %rdi
        leaq    0(,%rdi,8), %rax
        movq    t1+16(%rax), %rax
        movl    %esi, (%rax)
        ret
```

根据上述代码，填写函数 fun 的 C 语言代码：

```
void fun(int x, int y)
{
        (1) = (2) ;
        (3) = (4) ;
```

```
}
```

答案：

（1）t1[x].ii[y]

（2）y

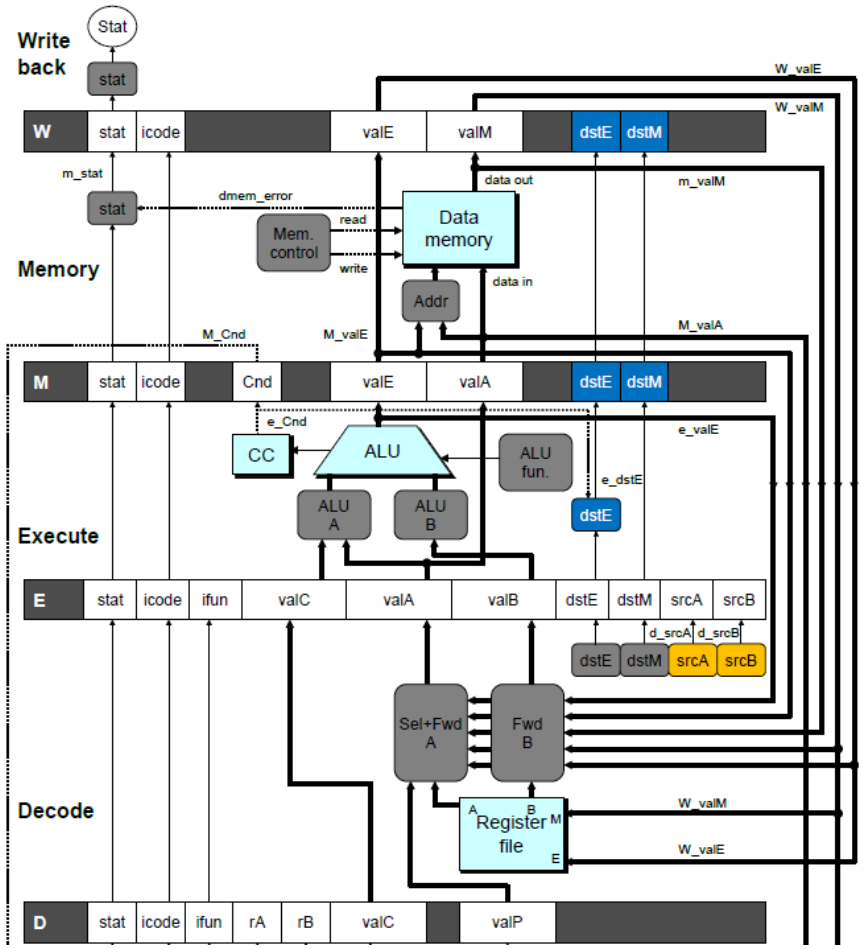
（3）*(t1[x].ip)

（4）y

得分

第四题（15 分）

这是一款 Y86-32 流水线处理器的结构图（局部），请以此为基础，依次回答下列问题。



1、该处理器设计采用了前递（forwarding）技术，一定程度上解决了数据相关的问题，在上图中体现在 Sel+FwdA 和 FwdB 部件上。前者输出的信号会存到流水线寄存器 E 的 valA 域（即 E_valA 信号），请补全该信号的 HCL 语言描述。

```
int E_valA = [
    D_icode in { ICALL, IJXX } :_____ ; # ① 答案：D_valP
    d_srcA == e_dstE :_____ ;# ② 答案：e_valE
    d_srcA == M_dstM :_____ ;# ③ 答案：m_valM
```

```
    d_srcA == M_dstE : M_valE       ;
    d_srcA == W_dstM : W_valM       ;
    …
];
```

2、如果在该处理器上运行下面的程序，每条指令在不同时钟周期所处的流水线阶段如下表所示。在这种情况下，哪条指令的执行结果会有错误？写出该指令的地址：0x01e___。（1 分）

```
demo1.ys
0x000: irmovl $128, %edx
0x006: irmovl $3, %ecx
0x00c: rmmovl %ecx, 0(%edx)
0x012: irmovl $10, %ebx
0x018: mrmovl 0(%edx),%eax
0x01e: addl %ebx, %eax
0x020: halt
```

1	2	3	4	5	6	7	8	9	10	11	12
F	D	E	M	W							
	F	D	E	M	W						
		F	D	E	M	W					
			F	D	E	M	W				
				F	D	E	M	W			
					F	D	E	M	W		
						F	D	E	M	W	

3、如需检测出这个情况，需要增加逻辑电路，用 HCL 语言表达如下：

E_icode in {IMRMOVL,IPOPL} && _____ in { _____ }

答案：E_icode in {IMRMOVL,IPOPL} && E_dstM in { d_srcA, d_srcB }，2 分，全对才得分

4、当新增的电路检测出这个情况后，应对各流水线寄存器进行不同的设置，以便在尽可能少影响性能的前提下解决该问题。请填写下表，可选的设置包括 normal/bubble/stall 三种。

F	D	E	M	W

答案：stall,stall,bubble,normal,normal。3 分，全对才得分

5、如果遇到下面程序代码所展示的情况，该处理器运行时仍然存在问题。因此，还需要新增检测电路。当新增的电路检测出这个情况后，应对各流水线寄存器进行不同的设置，以便在尽可能少影响性能的前提下解决该问题。请填写下表，可选的设置包括 normal/bubble/stall 三种。

```
demo2.ys
…
```

0x018: rmmovl %ecx, 0(%edx)
0x01e: irmovl $10, %ebx
0x024: popl %esp
0x026: ret

F	D	E	M	W

答案：stall,stall,bubble,normal,normal。3 分，全对才得分

得分

第五题（15 分）

Cache 为处理器提供了一个高性能的存储器层次框架。下面是一个 8 位存储器地址引用的列表（地址单位为字节，地址为 10 进制表示）：

3，180，43，2，191，88，190，14，181，44

1. (7 分)考虑如下 cache（S=2, E=2），每个 cache block 大小为 2 个字节。假设 cache 初始状态为空，替换策略为 LRU。请填补下表：
(Tag 使用二进制格式；Data 使用十进制格式，例：M[6-7]表示地址 6 和 7 对应的数据)

	V	Tag	Data	V	TAG	Data
SET 0	1			1		
SET 1	1			1		

共命中_____次（1 分），分别访问地址_____（地址用 10 进制表示，2 分）

解答：

答案：

	V	Tag	Data	V	TAG	Data
SET 0	1	101101 M[180-181]		1	001011 M[44-45]	
SET 1	1	000011 M[14-15]		1	101111 M[190-191]	

共命中 3 次

（表格上每空格 1 分，tag 和 data 都正确才得分；）

命中次数回答正确得 1 分；

分别为访问地址 2、190、181（三个地址完全正确得 2 分，答对两个得 1 分）

2. (5 分)现在有另外两种直接映射的 cache 设计方案 C1 和 C2，每种方案的 cache 总大小都为 8 个字节，C1 块大小为 2 个字节，C2 块大小为 4 个字节。假设从内存加载一次数据到 cache 的时间为 25 个周期，访问一次 C1 的时间为 3 个周期，访问一次 C2 的时间为 5 个周期。针对第一问的地址访问序列，哪一种 cache 的设计更好？（请分别给出两种 cache 访问第一问地址序列的总时间以及 miss rate）

答案：

Address	Binary address	C1 hit/miss	C2 hit/Miss
3	000000 11	M	M
180	101101 00	M	M
43	001010 11	M	M

2	000000 10	M	M
191	101111 11	M	M
88	010110 00	M	M
190	101111 10	H	H
14	000011 10	M	M
181	101101 01	H	M
44	001011 00	M	M

C1 更好。(1 分)

C1: miss rate = 8/10 = 80%, (1 分) total cycles = 8 * 25 + 10 * 3 = 230 (1 分)

C2: miss rate = 9/10 = 90%, (1 分) total cycles = 9 * 25 + 10 * 5 = 275 (1 分)

3. (3 分)现在考虑另外一个计算机系统。在该系统中，存储器地址为 32 位，并采用如下的 cache：

Cache datasize	Cache block size	Cache mode
32 KiB	8 Bytes	直接映射

此 cache 至少要占用_____Bytes. (datasize + (valid bit size + tag size) * blocks)

答案：

cache block 为 8 bytes，所以 b=3；

cache block 一共 32*1024 / 8 = 4096 个，又因为是直接映射，所以 s=12；于是 tag 位一共 t = 32 - s - t = 17。所以总大小为：

totalsize = datasize + (valid bit size + tag size) * blocks

= 32 * 1024 + (1 + 17) * 4096 / 8 = 41984 (bytes)

得分

第六题（15 分）

在 PIPE 处理器上运行如下 Y86 代码

```
.L1 mrmov (%eax) %ebx
    addl %ebx %ecx
    addl %ecx %eax
    xorl %ecx %edx
    jne .L1
    irmov $1 %eax
    irmov $1 %eax
```

1) 假设上述代码中的循环部分，一共执行了 N 遍后跳出循环，未采用数据前递（data forwarding），分支每次都预测正确，访存每次都命中缓存，命中缓存时访存需要 1 个周期。请问共需执行多少个周期？在下面空格处，各填入一个数字。

N * _11_ + _6_

答案：4+N * (5+3+3)+2，填充流水线 +4 ，一次 load-use 冒险 +3，一次 RAW 冒险+3，循环后 2 指令 +2

答案修订：N*12+5

4+(N-1)*(5+3+3+1)+(5+3+3)+2,填充流水线+4,一次 load-use 冒险+3(%ebx)，一次 RAW 冒险+3（%ecx），一次 RAW 冒险+1（%eax），循环后 2 指令+2，其中最后一次循环%eax 不会冒险。

2) 假设上述代码一共循环执行了 N 次后跳出循环，采用数据前递（data forwarding），分支每次都预测跳转（taken），访存每次都命中缓存，命中缓存时访存需要 1 个周期。请问共需执行多少个周期？

在下面空格处，各填入一个数字。

N * _6_ + _8_

答案：4+N * (5+1)+4 ，填充流水线 +4 ，一次 load-use 冒险 +1，最后一次预测错误+2，循环后 2 指令 +2

3) 假设上述代码一共循环执行了 N 次后跳出循环(N 为偶数)，采用数据前递(data forwarding)，分支每次都预测不跳转（not taken），数据访存时缓存命中率为 50%，指令访存全部命中缓存，命中缓存时访存需要 1 个周期，未命中缓存时访

存需要 3 个周期。请问共需执行多少个周期?

在下面空格处，各填入一个数字。

N * _9_ + _4_

答案：4+0.5*N * (5+1+2)+0.5*N*(5+3+2) +2-2，填充流水线 +4 ，hit 一次 load-use 冒险 +1,miss 一次 load-use 冒险 +3,预测错误+2,循环后 2 指令 +2 ，最后一次预测正确-2

（每小题 5 分。N 相乘的那个数 3 分，常数 2 分）