第一题 选择题（每小题 2 分，共 34 分）

（每小题有一个或多个正确答案）

1、变量 *x* 的值为 0x01234567，地址 &*x* 为 0x100；则该变量的值在 x86 和 Sun 机器内存中的存储排列顺序正确的是（　　）

选项	机器类型	地址			
		0x100	0x101	0x102	0x103
A	x86	67	45	23	01
	Sun	01	23	45	67
B	x86	76	54	32	10
	Sun	01	23	45	67
C	x86	01	23	45	67
	Sun	67	45	23	01
D	x86	01	23	45	67
	Sun	01	23	45	67

答案：A

考察大端、小端；同时 sun 是大端、x86 是小端

2、假设下列 int 和 unsigned 数均为 32 位，

int *x* = 0x80000000;

unsigned *y* = 0x00000001;

int *z* = 0x80000001;

以下表达式正确的是（　　）

A. (-*x*) < 0

B. (-1) > *y*

C. (*z*<<3) == (*z**8)

D. *y**24 == *z*<<5 - *z*<<3

答案:ABCD；考虑到运算符的优先顺序，选 ABC 也算对

A. int 中 0x80000000 的相反数还是自己

B. signed (-1) 和 unsinged y 比较，都按照 unsigned，所以强制类型转换后 （-1）很大

C: unsigned, signed 左移三位 = *8

D：应该是相等关系；signed 左移之后，和 unsigned y*24 相比都看成 unsigned

3、 对 $x = 1\frac{1}{8}$ 和 $y = 1\frac{3}{8}$ 进行小数点后两位取整（rounding to nearest even），结果正确的是（　　）

A. $1\frac{1}{4}$, $1\frac{1}{4}$　　B. 1, $1\frac{1}{4}$　　C. $1\frac{1}{4}$, $1\frac{1}{2}$　　D. 1, $1\frac{1}{2}$

答案：D

x= 1.00100　half way and down --> 1.00

y = 1.01100　half way and up-->1.10

4、在完成 Bomb Lab 的时候，通常先执行 gdb bomb 启动调试，然后执行 ___ explode_bomb 命令以防引爆炸弹，之后在进行其他必要的设置后，最后执行___命令以便开始执行程序。上述两个空格对应的命令是（　　）

A. st, ru　　B. br, go　　C. br, ru　　D. st, go

答案：c

说明：根据之前的讨论，出一道题目检查同学们是否自己做过 lab

5、已知函数 int x(int n) {　return　n*____; } 对应的汇编代码如下：

```
lea (%rdi, %rdi, 4), %rdi
lea (%rdi, %rdi, 1), %eax
retq
```

请问横线上的数字应该是（　　）

A. 4　　B. 5　　C. 2　　D. 10

答案：D

说明：此题目考察对于乘法的转换，难度较低，适合出选择题。还可以把乘法换成除法，就可以出大题或者简答题。

6、32 位 x86 计算机、Windows 操作系统下定义的一个 structure S 包含三个部分： double a, int b, char c, 请问 S 在内存空间中最多和最少分别能占据多少个字节（32 位 Windows 系统按 1、4、8 的原则对齐 char、int、double）？答：（　　）

A. 16, 13

B. 16, 16

C. 24, 13

D. 24, 16

答案：D 考虑对齐,windows double 按 8 字节对齐，最长 c, a, b，最短 a,b,c

7、x86 体系结构的内存寻址方式有多种格式，请问下列哪些指令是正确的：（　　）

A. movl　$34,　(%eax)

B. movl　(%eax),　%eax

C. movl　$23,　10(%edx, %eax)

D. movl　(%eax),　8(%ebx)

答案：ABC，寻址不支持内存到内存的访问

8、 x86 体系结构中，下面哪些选项是错误的？答：（　　）

A. leal 指令只能够用来计算内存地址

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. 判断两个寄存器中值大小关系，只需要 SF（符号）和 ZF（零）两个 conditional code

答案：ACD

9、下面对 RISC 和 CISC 的描述中，错误的是：（　　）

A. CISC 指令系统中的指令数目较多，有些指令的执行周期很长；而 RISC 指令系统中通常指令数目较少，指令的执行周期都较短。

B. CISC 指令系统中的指令编码长度不固定；RISC 指令系统中的指令编码长度固定，这样使得 RISC 机器可以获得了更短的代码长度。

C. CISC 指令系统支持多种寻址方式，RISC 指令系统支持的寻址方式较少。

D. CISC 机器中的寄存器数目较少，函数参数必须通过栈来进行传递；RISC 机器中的

寄存器数目较多，可以通过寄存器来传递参数，避免了不必要的存储访问。

答案：BD

10、下面对流水线技术的描述，正确的是：（　　）

A. 流水线技术不仅能够提高执行指令的吞吐率，还能减少单条指令的执行时间。

B. 不断加深流水线级数，总能获得性能上的提升。

C. 流水级划分应尽量均衡，吞吐率会受到最慢的流水级影响。

D. 指令间的数据相关可能会引发数据冒险，可以通过数据转发或暂停流水线来解决。

答案：CD

(11-13)、在教材所描述的流水线处理器（the PIPE processor）上分别运行如下四段Y86程序代码。请分析其中数据冒险的具体情况，并回答后续3个小题。

```
#Program 1:
mrmovl    8(%ebx), %edx
rmmovl    %edx, 16(%ecx)
```

```
#Program 2:
mrmovl    8(%ebx), %edx
nop
rmmovl    %edx, 16(%ecx)
```

```
#Program 3:
mrmovl    8(%ebx), %edx
nop
nop
rmmovl    %edx, 16(%ecx)
```

```
#Program 4:
mrmovl    8(%ebx), %edx
nop
nop
nop
rmmovl    %edx, 16(%ecx)
```

11、对于每段程序，请指出是否会因为数据冒险导致流水线停顿（Stall）。

Program 1：（　　），Program 2：（　　），Program 3：（　　），Program 4：（　　）；

A. Stall　　B. No-Stall

答案：　A, B, B, B

12、对于每段程序，请指出流水线处理器内是否会产生数据转发（Forwarding）。

Program 1：（　　），Program 2：（　　），Program 3：（　　），Program 4：（　　）；

A. Forwarding　　B. No-Forwarding

答案: A,A,A,B

13、对于每段程序，请指出流水线处理器内使用哪个信号进行数据转发，如果不进行数据转发，则用none表示。

Program 1：（　　），Program 2：（　　），Program 3：（　　），Program 4：（　　）；

A. m_valM　　B. W_valM　　C. none

答案: A,A,B,C

14、下面哪些选项是错误的？答：（　　）

A. 同一个任务采用时间复杂度为 O(logN)算法一定比采用复杂度为 O(N)算法的执行时间短

B. 编译器进行程序优化时，总是可以使用算数结合律来减少计算量

C. 增大循环展开（loop unrolling）的级数，有可能降低程序的执行性能（即增加执行时间）

D. 分支预测时，“总是预测不跳转”（branch not taken）一定比“总是预测跳转”（branch taken）

预测准确率高

答案：ABD

15、以下哪些程序优化编译器总是可以自动进行？（假设 int i, int j, int A[N], int B[N], int m 都是局部变量，N 是一个整数型常量，int foo(int) 是一个函数）答：（　　）

	优化前	优化后

A.	for (j = 0 ; j < N ; j ++)
	m + = i*N*j;
		int temp = i*N;
		for (j= 0 ; j < N ; j ++)
		m + = temp * j;

B.	for (j = 0 ; j < N ; j ++)
	B[i] *= A[j];
		int temp = B[i];
		for (j= 0 ; j < N ; j ++)
		temp *= A[j];
		B[i] = temp;

C.	for (j = 0 ; j < N ; j ++) m = (m + A[j]) + B[j];	for (j = 0 ; j < N ; j ++) m = m + (A[j] + B[j]);

D.	for (j = 0 ; j < foo(N) ; j ++)
	m ++;
		int temp = foo(N);
		for (j= 0 ; j < temp ; j ++)
		m ++;

答案：AC

16、如果直接映射高速缓存大小是 4KB，并且块（block）大小为 32 字节，请问它每组（set）有多少行（line）？答：（　　）

A. 128　　B. 64　　C. 32　　D. 1

答案：D

17、关于局部性（locality）的描述，不正确的是：（　　）

A. 数组通常具有很好的时间局部性

B. 数组通常具有很好的空间局部性

C. 循环通常具有很好的时间局部性

D. 循环通常具有很好的空间局部性

答案：A

第二题（8 分）

1）判断下表中每一行表达式对或错。如果错，请举出反例或简要说明原因（每行 1 分）

int *x*, *y*;

unsigned *u*, *v*;

	True or false	原因或举出反例
if *x*<0, then *x**2 <0		
u<= -1		
if *x*> *y*, then -*x* < -*y*		
if *u*> *v*, then -*u* > -*v*		

答案

	True or false	原因或举出反例
if *x*<0, then *x**2 <0	F	X= -2^{w}-1
u<= -1	T	-1 作为无符号来比大于 u
if *x*> *y*, then -*x* < -*y*	F	X=0, y= -2^{w}-1
if *u*> *v*, then -*u* > -*v*	F	U=2, v=1

2）请按 IEEE 浮点标准的单精度浮点数表示下表中的数值，首先写出形如$(-1)^{s} \times M \times 2^{E}$ 的表达式，然后给出十六进制的表示。（每格 1 分）

注：单精度浮点数的字段划分如下：

符号位（s）：1-bit；阶码字段（exp）：8-bit；小数字段（frac）：23-bit；偏置值（bias）：127。

Value	$(-1)^{s} \times M \times 2^{E}$，1<=M<2	Hex representation
$-1\frac{1}{2}$		
2^{-149}		

答案

Value	$(-1)^{s}$ x M x 2^{E} 1<=M<2	Hex representation
$-1\frac{1}{2}$	(-1) x 1.1 x 2^{0}	0xBFC00000
2^{-149}	1.0 x 2^{-149}	0x00000001

第三题 （11 分）

阅读下面的 C 代码：

```
/*
 * Copyright (C) 2013 Davidlohr Bueso <davidlohr.bueso@hp.com>
 *
 *  Based on the shift-and-subtract algorithm for computing integer
 *  square root from Guy L. Steele.
 */
/**
 * int_sqrt - rough approximation to sqrt
 * @x: integer of which to calculate the sqrt
 *
 * A very rough approximation to the sqrt() function.
 */
unsigned long int_sqrt(unsigned long x)
{
        unsigned long b, m, y = 0;

        if (x <= 1)
                return x;

        m = 1UL << (BITS_PER_LONG - 2);
        while (m != 0) {
                b = y + m;
                y >>= 1;

                if (x >= b) {
                        x -= b;
                        y += m;
                }
                m >>= 2;
        }

        return y;
}
```

1）在 64 位的机器上 BITS_PER_LONG 的定义为 long 类型的二进制位数，它是多少位？

2）填写下面反汇编中的缺失的内容：

```
<int_sqrt>:
   4004c4:      push   %rbp
   4004c5:      mov    %rsp,%rbp
   4004c8:      mov    %rdi,-0x28(%rbp)

   4004cc:      movq   (1)_____,-0x8(%rbp)
   4004d4:      cmpq   $0x1,-0x28(%rbp)

   4004d9:      ja     (2)_____ <int_sqrt+??>
   4004db:      mov    -0x28(%rbp),%rax

   4004df:      jmp    (3)_____ <int_sqrt+??>
   4004e1:      movl   $0x0,-0x10(%rbp)

   4004e8:      movl   (4)_____,-0xc(%rbp)

   4004ef:      jmp    (5)_____ <int_sqrt+??>
   4004f1:      mov    -0x10(%rbp),%rax
   4004f5:      mov    -0x8(%rbp),%rdx
```

```
4004f9:    lea     (6)                ,%rax
4004fd:    mov     %rax,-0x18(%rbp)
400501:    shrq    -0x8(%rbp)
400505:    mov     -0x28(%rbp),%rax
400509:    cmp     -0x18(%rbp),%rax

40050d:    jb      (7)                <int_sqrt+??>
40050f:    mov     -0x18(%rbp),%rax
400513:    sub     %rax,-0x28(%rbp)
400517:    mov     -0x10(%rbp),%rax
40051b:    add     %rax,-0x8(%rbp)
40051f:    shrq    (8)                ,-0x10(%rbp)
400524:    cmpq    $0x0,-0x10(%rbp)
400529:    jne     (9)                <int_sqrt+??>
40052b:    mov     -0x8(%rbp),    (10)
40052f:    leaveq
400530:    retq
```

答案：

1、 答：64

2、

```
<int_sqrt>:
   4004c4:    push    %rbp
   4004c5:    mov     %rsp,%rbp
   4004c8:    mov     %rdi,-0x28(%rbp)
   4004cc:    movq    $0x0,-0x8(%rbp)
   4004d4:    cmpq    $0x1,-0x28(%rbp)
   4004d9:    ja      4004e1 <int_sqrt+0x1d>
   4004db:    mov     -0x28(%rbp),%rax
   4004df:    jmp     40052f <int_sqrt+0x6b>
   4004e1:    movl    $0x0,-0x10(%rbp)
   4004e8:    movl    $0x40000000,-0xc(%rbp)
   4004ef:    jmp     400524 <int_sqrt+0x60>
   4004f1:    mov     -0x10(%rbp),%rax
   4004f5:    mov     -0x8(%rbp),%rdx
   4004f9:    lea     (%rdx,%rax,1),%rax
   4004fd:    mov     %rax,-0x18(%rbp)
   400501:    shrq    -0x8(%rbp)
   400505:    mov     -0x28(%rbp),%rax
   400509:    cmp     -0x18(%rbp),%rax
   40050d:    jb      40051f <int_sqrt+0x5b>
   40050f:    mov     -0x18(%rbp),%rax
   400513:    sub     %rax,-0x28(%rbp)
   400517:    mov     -0x10(%rbp),%rax
   40051b:    add     %rax,-0x8(%rbp)
   40051f:    shrq    $0x2,-0x10(%rbp)
   400524:    cmpq    $0x0,-0x10(%rbp)
   400529:    jne     4004f1 <int_sqrt+0x2d>
   40052b:    mov     -0x8(%rbp),%rax
   40052f:    leaveq
   400530:    retq
```

第四题（10 分）

阅读下面的汇编代码：

```
<f>:
   4004c4:          push    %rbp
   4004c5:          mov     %rsp,%rbp
   4004c8:          sub     $0x10,%rsp
   4004cc:                  mov     %edi,-0x4(%rbp)
   4004cf:                  cmpl    $0x1,-0x4(%rbp)
   4004d3:          ja      4004dc <f+0x18>
   4004d5:          mov     $0x1,%eax
   4004da:          jmp     40052d <f+0x69>
   4004dc:          mov     -0x4(%rbp),%eax
   4004df:                  and     $0x1,%eax
   4004e2:          test    %eax,%eax
   4004e4:          jne     4004f5 <f+0x31>
   4004e6:          mov     0x200440(%rip),%eax        # 60092c <x.1604>
   4004ec:                  add     $0x1,%eax
   4004ef:                  mov     %eax,0x200437(%rip)        # 60092c <x.1604>
   4004f5:                  mov     -0x4(%rbp),%eax
   4004f8:                  and     $0x1,%eax
   4004fb:                  test    %al,%al
   4004fd:                  je      40050e <f+0x4a>
   4004ff:                  mov     0x20042b(%rip),%eax        # 600930 <y.1605>
   400505:          add     $0x1,%eax
   400508:          mov     %eax,0x200422(%rip)        # 600930 <y.1605>
   40050e:          mov     -0x4(%rbp),%eax
   400511:          sub     $0x1,%eax
   400514:          mov     %eax,%edi
   400516:          callq   4004c4 <f>
   40051b:          mov     0x20040f(%rip),%edx        # 600930 <y.1605>
   400521:          lea     (%rax,%rdx,1),%edx
   400524:          mov     0x200402(%rip),%eax        # 60092c <x.1604>
   40052a:          lea     (%rdx,%rax,1),%eax
   40052d:          leaveq
   40052e:          retq
```

1）程序

```
main()
{
    unsigned int n;
    for (n=1; n< 4; n++) {
        printf("f(%d) = %x\n", n, f(n));
    }

}
```

的运行结果为：f(1)=1，f(2)=4e，f(3)=9f，请填写 f 函数所需要的内容（每空 1 分）：

```
#define N     (1)_____
#define M     (2)_____

struct P1 {char c[N]; char *d[N]; char e[N]; } P1;
struct P2 {int i[M]; char j[M]; short k[M]; } P2;

unsigned int f(unsigned int n)
{
    (3)_____ unsigned int x = sizeof(P1);
    (4)_____ unsigned int y = sizeof(P2);
```

```
    if (   (5)                          )
        return 1;

    if (   (6)                          )
        x++;

    if (   (7)                          )
        y++;

    return   (8)                        ;
}
```

2、程序

```
main()
{
    printf("%x, %x\n", f(2), f(2));
}
```

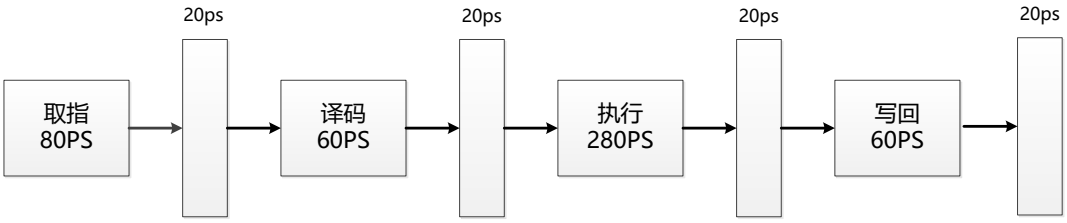
的运行结果为：（2 分）

1、答案：

```
#define N     3
#define M     5

struct P1 {char c[N]; char *d[N]; char e[N]; } P1;
struct P2 {int i[M]; char j[M]; short k[M]; } P2;

unsigned int f(unsigned int n)
{
    static unsigned int x = sizeof(P1);
    static unsigned int y = sizeof(P2);

    if (n<=1)
        return 1;

    if ((n & 1) == 0)
        x++;

    if ((n & 1) == 1)
        y++;

    return f(n-1) + (y) +    (x);
}
```

2、答案：4f, 4e　　（回答 4e, 4f 给一半的分）

第五题（9 分）



在“取指-译码-执行-写回”的四级流水线中，各流水级的工作内容和延迟如上图所示，寄存器的延迟也已标出。数据和指令分别存放在不同的存储器中。Cycle N 写入寄存器文件的数据 Cycle N+1 才可读出。请问：

1）若不考虑流水线填充和清空时间，请计算该处理器的吞吐率。（1 分）

2）若将该处理器改造为单周期处理器（SEQ），请计算 SEQ 处理器的吞吐率。（1 分）

3）在上述流水线中，执行阶段包含了访问数据存储的时间。对于如下的 Y86 程序段，指令间存在哪些数据相关（dependence），会引起哪些数据冒险（hazard）？（5 分）

```
Prog:
irmovl$128, %edx            #instr1
irmovl$3, %ecx              #instr2
rmmovl      %ecx, 0(%edx)   #instr3
irmovl$10, %ebx         #instr4
mrmovl      0(%edx), %eax   #instr5
addl        %ebx, %eax          #instr6
```

4）以上的数据冒险，可以通过转发（forward）的方法解决。请结合上述程序代码和流水线结构图逐个说明解决方案。（2 分）

答案：

1）1000/(280 + 20) = 1000/300 = 3.33GIPS

2）1000/(80 +60 +280 + 60 +20) = 1000/ 500 = 2 GIPS

3）

```
Prog：
irmovl$128, %edx            ; 1、dx= 128
irmovl$3, %ecx              ; 2、cx= 3
rmmovl      %ecx, 0(%edx)   ; 3、[dx] = cx
irmovl$10, %ebx         ; 4、bx= 10
mrmovl      0(%edx), %eax   ; 5、ax= [dx]
addl        %ebx, %eax          ; 6、bx = bx + ax
```

相关：1-3，2-3, 1-5，4-6，5-6

冒险：1-3，2-3，4-6，5-6

4）2-3，5-6：执行到译码的转发通路解决；1-3,4-6：写回到译码的转发通路解决。

第六题（9 分）

请分析Y86 ISA中定义的两条指令（cmovXX、call）和一条新加入Y86 ISA的IA32指令（decl：将操作数减1）。若在教材所描述的SEQ处理器上执行这些指令，请按下表填写每个阶段进行的操作。如果在某一阶段没有任何操作，请填写none指明。

注1、所用到的指令编码为：

cmovXX rA, rB

2	fn	rA	rB

call Dest

8	0	Dest

decl rA

C	0	rA	F

注2、需说明的信号包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP；the register file R[], data memory M[], Program counter PC, condition codes CC。

（每格0.5分）

Stage	cmovXX rA, rB	call Dest	decl rA
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2	icode:ifun ← M_1[PC] valC ← M_4[PC+1] valP ← PC+5	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2
Decode	valA ← R[rA]	valB ← R[%esp]	valA ← R[rA]
Execute	valE ← 0+valA Cnd ← Cond(CC,ifun) （也可以写Set CC）	valE ← valB+(-4)	valE ← valA+(-1) Cnd ← Cond(CC,ifun) （也可以写Set CC）
Memory	none	M_4[valE] ← valP	none
Write back	if(Cnd) R[rB] ← valE	R[%esp] ← valE	R[rA] ← valE
PC update	PC ← valP	PC ← valC	PC ← valP

第七题（10 分）

已知如下的汇编程序实现了函数 transform(char* src, char* tgt, char delta)

```
transform:
   jmp L2
L1:
   add %edx, %eax
   add $1, $rdi
   mov %al, (%rsi)
   add $1, $rsi
L2:
   movzbl (%rdi), %eax
   test %al, %al
   jne L1
   movb $0, (%rsi)
   retq
```

参考信息：64 位指令集中传递前三个参数分别使用寄存器%rdi, %rsi 和%rdx

1）写出 transform 函数对应的 C 语言版本（2 分）

2）假设读写访存指令延迟为 20 个时钟周期，其他指令延迟为 2 个时钟周期，所有分支预测都成功。同时 CPU 包含足够多的部件来实现指令集并行，那么在最理想情况下 CPE 最低应该是多少（2 分）？为什么（2 分）？

3）已知 src 对应字符串中每个字符 c 都满足 0<c<=80 且 0<=delta<=5。通过下面的改写，可以把 transform 程序 CPE 的理论下限降低一半，请填空。假设程序运行在小端法机器上。（每空 1 分）

```
void transform(char* src, char* tgt, char delta) {
   short x = _______________;
   while(*src && _____________) {
      *(short*)tgt = *(short*)src + x;
      src += 2;
      tgt += 2;
   }
   *(short*)tgt = ___________ ? *(short*)src + delta : *(short*)tgt &_____________;
}
```

答案：

（1） {while(*src) *tgt++ = *src++ + delta; *tgt = 0;}

（2） 42

每次循环的关键路径为 读内存、做加法、写内存，该路径需要 42 个时钟周期。

本题陷阱：同学可能会受书上的例子误导认为做加法可以和下一个时钟周期的读内存并行，使得 CPE 降到 40，但实际上因为 src 和 tgt 指向的位置可能重叠，我们不能把下一次迭代的读操作移动到这一次的写操作之前。

（3） (((short) delta) << 8) + delta

*(src+1)

0xFF00

*src

第八题（9 分）

假设存在一个能够存储四个数据的 Cache，每一个 line 的长度（B）为 2 字节。假设内存空间的地址一共是 32 字节，既内存空间地址长度一共是 5 个比特：从 0（00000）到 31（11111），一共有 8 个数据读取操作，每个操作的地址按顺序如下所示（单位是字节），数据替换采用 LRU（least recently used）策略。

数据访问地址： 1 -> 4 -> 17 –> 2 –> 8 -> 16 –> 9->0

1） 如果 Cache 的结构是 directed mapped（S=4， E=1），如下图所示，请在下图空白处填入，访问上述数据序列访问后 Cache 的状态。（注：TAG 使用二进制格式，V=1 代表数据有效，用[A-B]表示地址 A 到 B 之间对应的数据）（4 分）

V	TAG	DATA
1	00	M[0-1]
1	00	M[2-3]
1	00	M[4-5]
0		

2） 如果 cache 的结构如下图所示既(S = 2, E = 2)， 请填入访问后的状态 （2 分）

V	TAG	DATA	V	TAG	DATA
1	000	M[0-1]	1	010	M[8-9]
1	000	M[2-3]	0		

在这种情况下，数据访问一共产生了多少次 Miss ____6____ （1 分）

（顺序 M-> M-> M-> M-> M-> H->H-> M）

3）如果 cache 的结构变成 （S=1， E=4），最终存储在 Cache 里面的数据有那些（注：只需要填写数据部分，顺序不限）？

_____， _____， _____， _____ （2 分）

M[0-1]， M[8-9]， M[16-17]， M[2-3]

说明：1 -> 4 -> 17 –> 2 –> 8 -> 16 –> 9->0 （后四个不重复数据）