# 考试科目名称　计算机系统基础　（A 卷）

2015—2016 学年第 1 学期　教师　路通 苏丰 唐杰　考试方式：开卷

系（专业）　计算机科学与技术　　年级＿＿＿＿＿＿　　班级＿＿＿＿

学号＿＿＿＿＿＿＿＿＿＿　　姓名＿＿＿＿＿＿　　成绩＿＿＿＿＿＿

| 题号 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 十一 | 十二 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 分数 | | | | | | | | | | | | | |

一、简答题（每题 4 分，共 20 分）

1、DMA 方式实现系统 I/O 的特点是什么?

DMA 方式：I/O 设备和内存间直接通过 DMA 硬件批量传输数据。

2、什么是陷阱指令? 它的作用是什么?

用户程序需通过自陷指令向操作系统提供系统服务对应的编号来使用内核提供的相应服务。

3、解释在程序链接过程中需要进行重定位的原因?

4、请列举至少五种会导致异常控制流的原因。

如溢出、缺页、越界、非法指令、除数为 0、堆栈溢出、断点设置、打印机缺纸、电源掉电、硬件故障、采用时间到等。

5、什么是快表?

活跃页表在高速缓存中的镜像。

二、一个 C 语言程序有两个源文件：main.c 和 test.c，它们的内容如下图所示

```
/* main.c */
1    #include <stdio.h>
2
3    int add(int a, int b);
4    int a[2]={-1, 3};
5    float b[2] = {-1.5, -0.75}
6    void main( )
7    {
8          add(a[0],b[1]);
9          unsigned char val=a[0];
10         printf("val=%d\n",val);
10   }
```

```
/* test.c */
1
2
3    int val=0;
4    int add(int a, int b)
5    {
6         int i=-1;
7         if(a>0)
8            i= a+b;
9         return i;
10   }
```

请回答如下问题（提示：IA-32 为小端方式，字长为 32 位，即 sizeof(int)=4，虚拟地址空间中的只读数据和代码段、可读写数据段都按 4KB 边界对齐）

1、 请简述从 C 语言源程序到可执行文件 test 的转换需要经过哪些步骤？ **（4 分）**

**略**

2、 如图所示程序片段，已知数组 a 首址为 0x080496dc，则 0x080496e0 到 0x080496e3 每个单元的内容依次是什么？假设数组 b 的首地址是 0x080496e4，则 0x080496e8 到 0x080496eb 每个单元的内容依次是什么？变量 val 的机器值是多少？printf 语句打印出的值是多少？ **（8 分）**

**参考答案**

**在 0x080496e0 到 0x080496e3 这 4 个单元中存放的是 3，3 =11B， 在机器中的 32 位原码表示为 00000003H。因为 IA-32 是小端方式，因此，在 0x080496e0 到 0x080496e3 这 4 个单元的内容依次为：03H、00H、00H、00H。（2 分）**

**在 0x080496e8 到 0x080496eb 这 4 个单元中存放的是浮点数-0.75，3 = BF400000H，因为 IA-32 是小端方式，因此，在 0x080496e8 到 0x080496eb 这 4 个单元的内容依次为：00H、00H、40H、BFH。（2 分）**

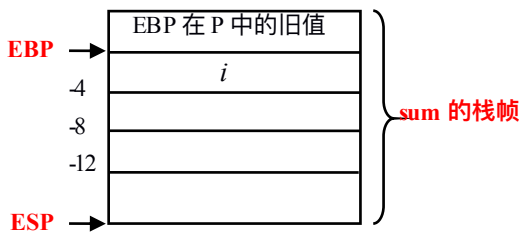
**Val 的机器值是 FFH，（2 分）**

**prinft 输出的值是 255 （2 分）**

3、根据下图 add 函数反汇编结果画出其栈帧，要求分别用 EBP 和 ESP 标示栈帧底部和顶部并标出 i 的位置。**（4 分）**

```
0804841c <add>:
804841c:   55                      push   %ebp
804841d:   89 e5                   mov    %esp,%ebp
804841f:   83 ec 10                sub    $0x10,%esp
8048422:   c7 45 fc ff ff ff ff    movl   $0xffffffff,-0x4(%ebp)
8048429:   83 7d 08 00             cmpl   $0x0,0x8(%ebp)
804842d:   7e 0b                   jle    804843a <add+0x1e>
804842f:   8b 45 0c                mov    0xc(%ebp),%eax
8048432:   8b 55 08                mov    0x8(%ebp),%edx
8048435:   01 d0                   add    %edx,%eax
8048437:   89 45 fc                mov    %eax,-0x4(%ebp)
804843a:   8b 45 fc                mov    -0x4(%ebp),%eax
804843d:   c9                      leave
804843e:   c3                      ret
804843f:   90                      nop
```

参考答案




3、 如上图所示，cmpl 指令的执行将会影响 EFLAGS 寄存器中哪些常用标志？根据当前程序的数据，add 函数中 cmpl 指令的执行结果将如何影响下条 jle 指令？ **（10 分）**

**参考答案**

**cmpl 指令通过做减法来生成标志信息，其执行将影响 EFLAGS 寄存器中的 OF、CF、ZF 和 SF 这几位条件标志位。（4 分）**

**因为 a=-1，cmpl 指令中的操作数 M[R（%ebp）+8]实际上就是 a，因此，cmpl 指令实际上是在以下电路中实现“-1 减 0”的功能。如果“-1<=0”,则跳转到 0x804843a 处执行，否则执行 0x804842f 语句。（6 分）**

5、地址 0x804842f 处的 mov 指令中，源操作数采用什么寻址方式？0x8048435、0x8048437 处的汇编代码对应的 c 代码是哪一句？leave 指令和 ret 指令执行的操作是什么？ **（8 分）**

**参考答案**

**基址+位移　（2 分）**

**i = a + b；　（2 分）**

**leave 执行的是 movl %ebp, %esp**

**popl %ebp　（2 分）**

**ret 执行的是　1）取出返回地址，2）将返回地址送入 EIP 中　（2 分）**

三、有一 C 程序由如下两个模块 main.c 和 guess.c 组成（左边是 C 源程序，右边是其可重定位目标文件的反汇编代码）。回答下列问题：

程序模块 main.o：

```
#include <stdio.h>

unsigned int answer = 1;
int value = 1;

void main()
{
    int value;

    do {
        printf("Input a number: ");
        scanf("%d", &value);

        guess(value);
    } while (answer != 0);
}
```

```
00000000 <main>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 e4 f0                and    $0xfffffff0,%esp
   6:   83 ec 20                sub    $0x20,%esp
   9:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  10:   e8 fc ff ff ff          call   11 <main+0x11>
  15:   8d 44 24 1c             lea    0x1c(%esp),%eax
  19:   89 44 24 04             mov    %eax,0x4(%esp)
  1d:   c7 04 24 11 00 00 00    movl   $0x11,(%esp)
  24:   e8 fc ff ff ff          call   25 <main+0x25>
  29:   8b 44 24 1c             mov    0x1c(%esp),%eax
  2d:   89 04 24                mov    %eax,(%esp)
  30:   e8 fc ff ff ff          call   31 <main+0x31>
  35:   a1 00 00 00 00          mov    0x0,%eax
  3a:   85 c0                   test   %eax,%eax
  3c:   75 cb                   jne    9 <main+0x9>
  3e:   c9                      leave
  3f:   c3                      ret
```

程序模块 guess.o：

```
#include <stdio.h>

int answer;

void guess( int value )
{
        static int seed = 2015;

        if (value == seed) {
                answer = 0;
                printf("Bingo!\n");
                return;
        }

        if (value > seed) {
                answer = 1;
                printf("Less!\n");
        }
        else {
                answer = -1;
                printf("More!\n");
        }

        seed = (value + seed) >> 1;
}
```

```
00000000 <guess>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 18                sub    $0x18,%esp
   6:   a1 00 00 00 00          mov    0x0,%eax
   b:   39 45 08                cmp    %eax,0x8(%ebp)
   e:   75 18                   jne    28 <guess+0x28>
  10:   c7 05 00 00 00 00 00    movl   $0x0,0x0
  17:   00 00 00
  1a:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  21:   e8 fc ff ff ff          call   22 <guess+0x22>
  26:   eb 4a                   jmp    72 <guess+0x72>
  28:   a1 00 00 00 00          mov    0x0,%eax
  2d:   39 45 08                cmp    %eax,0x8(%ebp)
  30:   7e 18                   jle    4a <guess+0x4a>
  32:   c7 05 00 00 00 00 01    movl   $0x1,0x0
  39:   00 00 00
  3c:   c7 04 24 07 00 00 00    movl   $0x7,(%esp)
  43:   e8 fc ff ff ff          call   44 <guess+0x44>
  48:   eb 16                   jmp    60 <guess+0x60>
  4a:   c7 05 00 00 00 00 ff    movl   $0xffffffff,0x0
  51:   ff ff ff
  54:   c7 04 24 0d 00 00 00    movl   $0xd,(%esp)
  5b:   e8 fc ff ff ff          call   5c <guess+0x5c>
  60:   8b 15 00 00 00 00       mov    0x0,%edx
  66:   8b 45 08                mov    0x8(%ebp),%eax
  69:   01 d0                   add    %edx,%eax
  6b:   d1 f8                   sar    %eax
  6d:   a3 00 00 00 00          mov    %eax,0x0
  72:   c9                      leave
  73:   c3                      ret
```

1）　填写下表有关程序中符号的属性信息。注意：如相应表项不适用，请填写 n/a。 **（5 分）**

| 符号引用 | 符号定义在何可 | 类型 | 位于可执行目标 |
|---|---|---|---|

| | 重定位模块? | (global/local/auto) | 文件的何节? |
|---|---|---|---|
| guess（main 模块中） | | | |
| answer（guess 模块中） | | | |
| value（main 模块中） | | | |
| value（main 函数中） | | | |
| seed（guess 模块中） | | | |

2） 指出 main.o 可重定位目标模块中所有需要重定位之处的偏移量及其目标符号名。**（7 分）**

四、假设内存采用字节编址，每次内存访问的字大小为 1 字节，物理地址长度为 12 比特，采用 4 路组相联 cache，包含共 32 个 cache 行，块大小为 2 字节。下表显示当前 cache 中的内容（以十六进制表示）。回答下列问题：

4-way Set Associative Cache

| Index | Tag | Valid | Byte 0 | Byte 1 | Tag | Valid | Byte 0 | Byte 1 | Tag | Valid | Byte 0 | Byte 1 | Tag | Valid | Byte 0 | Byte 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1C | 0 | 05 | 20 | 01 | 0 | 13 | A4 | F0 | 0 | DF | 3A | 7F | 1 | C1 | 3F |
| 1 | 3B | 0 | F3 | 66 | E0 | 0 | 95 | 1F | 3B | 1 | 49 | B5 | 37 | 1 | 70 | AC |
| 2 | 0F | 0 | 79 | AC | AF | 1 | 96 | FF | 99 | 0 | 22 | B1 | 3A | 1 | 5F | 00 |
| 3 | 29 | 0 | 38 | 68 | 87 | 0 | F2 | 29 | 7D | 1 | 64 | 39 | 8B | 1 | AE | 34 |
| 4 | EA | 1 | D3 | DE | CC | 1 | AA | 17 | 8A | 0 | 2C | 67 | 18 | 1 | DB | B4 |
| 5 | A7 | 1 | 05 | 3C | AB | 1 | A4 | 04 | E3 | 0 | EE | D2 | 01 | 0 | 04 | E2 |
| 6 | F3 | 1 | 3C | A4 | 3D | 1 | DB | 8F | 4A | 1 | A5 | 0C | D9 | 1 | 3A | 0D |
| 7 | 80 | 1 | AB | 8D | 2B | 0 | 0E | 35 | 49 | 1 | 70 | 19 | 00 | 0 | 57 | 60 |

1） 给出物理地址各位的划分，即块内偏移、行索引、标记分别对应物理地址中的哪些位？ **（4 分）**

2） 在下列表格中分别给出访问物理地址 0x7D7、0x01A、0x3DD 时 cache 相关信息**（6 分）**：

| 物理地址 | 0x7D7 | 0x01A | 0x3DD |
|---|---|---|---|
| 块内偏移 | | | |
| Cache 行索引 | | | |
| Cache 标记 | | | |
| Cache 命中/缺失? | | | |
| 返回字节值(缺失时填 n/a) | | | |

五、考虑如下矩阵求和函数，其中 src 矩阵的存储开始于地址 0，dst 矩阵的存储紧接其后。假设使用数据区大小为 32KB 的直接映射 cache，块大小为 32 字节，size(int)==4。Cache 初始为空，局部变量 i、j、sum 保存于寄存器中。请计算如下不同设置情况下运行函数引起的 cache 缺失率（需给出计算过程或说明）。**（9 分）**

```
void sum1_matrix(int dest[ROWS][COLS], int src[ROWS][COLS])
{
    int i, j, sum;
    for (i=0; i<ROWS; i++)
        for (j=0; j<COLS; j++)
            sum += src[i][j] + dest[ROWS-1-i][j];
}
```

1） ROWS=64、COLS=64：

2） ROWS=128、COLS=64：

3） ROWS=64、COLS=96，同时函数改为如下形式：

```
void sum2_matrix(int dest[ROWS][COLS], int src[ROWS][COLS])
{
    int i, j, sum;
    for (j=0; j<COLS; j++)
        for (i=0; i<ROWS; i++)
            sum += src[i][j] + dest[ROWS-1-i][j];
}
```

六、假定一个计算机系统中有一个TLB和一个L1 Data Cache。该系统按字节编址，虚拟地址16位，物理地址12位，页大小为128B；TLB采用4路组相联方式，共有16个页表项；L1 Data Cache采用直接映射方式，块大小为4B，共16行。在系统运行到某一时刻时，TLB、页表和L1 Data Cache中的部分内容如下（10分）：

| 组号 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 03 | – | 0 | 09 | 0D | 1 | 00 | – | 0 | 07 | 02 | 1 |
| 1 | 13 | 2D | 1 | 02 | – | 0 | 04 | – | 0 | 0A | – | 0 |
| 2 | 02 | – | 0 | 08 | – | 0 | 06 | – | 0 | 03 | – | 0 |
| 3 | 07 | – | 0 | 63 | 0D | 1 | 0A | 34 | 1 | 72 | – | 0 |

(a)　TLB（4 路组相联）：4 组、16 个页表项

| 虚页号 | 页框号 | 有效位 |
|---|---|---|
| 00 | 08 | 1 |
| 01 | 03 | 1 |
| 02 | 14 | 1 |
| 03 | 02 | 1 |
| 04 | – | 0 |
| 05 | 16 | 1 |
| 06 | – | 0 |
| 07 | 07 | 1 |
| 08 | 13 | 1 |
| 09 | 17 | 1 |
| 0A | 09 | 1 |
| 0B | – | 0 |
| 0C | 19 | 1 |
| 0D | – | 0 |
| 0E | 11 | 1 |
| 0F | 0D | 1 |

(b) 部分页表：（开始 16 项）

| 行索引 | 标记 | 有效位 | 字节 3 | 字节 2 | 字节 1 | 字节 0 |
|---|---|---|---|---|---|---|
| 0 | 19 | 1 | 12 | 56 | C9 | AC |
| 1 | – | 0 | – | – | – | – |
| 2 | 1B | 1 | 03 | 45 | 12 | CD |
| 3 | – | 0 | – | – | – | – |
| 4 | 32 | 1 | 23 | 34 | C2 | 2A |
| 5 | 0D | 1 | 46 | 67 | 23 | 3D |
| 6 | – | 0 | – | – | – | – |
| 7 | 16 | 1 | 12 | 54 | 65 | DC |
| 8 | 24 | 1 | 23 | 62 | 12 | 3A |
| 9 | – | 0 | – | – | – | – |
| A | 2D | 1 | 43 | 62 | 23 | C3 |
| B | – | 0 | – | – | – | – |
| C | 12 | 1 | 76 | 83 | 21 | 35 |
| D | 16 | 1 | A3 | F4 | 23 | 11 |
| E | 33 | 1 | 2D | 4A | 45 | 55 |
| F | – | 0 | – | – | – | – |

(c) L1 Data Cache：直接映射，共 16 行，块大小为 4B

请问（假定图中数据都为十六进制形式）：

（1）虚拟地址中哪几位表示虚拟页号？哪几位表示页内偏移量？虚拟页号中哪几位表示 TLB 标记？哪几位表示 TLB 组索引？

（2）物理地址中哪几位表示物理页号？哪几位表示页内偏移量？

（3）CPU 从虚拟地址 067AH 中取出的值为多少？说明 CPU 读取地址 067AH 中内容的过程。

答案：

（1）16 位虚拟地址中低 7 位为页内偏移量，高 9 位为虚页号；虚页号中高 7 位为 TLB 标记，低 2 位为 TLB 组索引。

（2）12 位物理地址中低 7 位为页内偏移量，高 5 位为物理页号。

（3）地址 067AH=0000 0110 0111 1010B，所以，虚页号为 0000011 00B，映射到 TLB 的第 00 组，将 0000011B=03H 与 TLB 第 0 组的四个标记比较，虽然和其中一个相等，但对应的有效位为 0，其余都不等，所以 TLB 缺失，需要访问主存中的慢表。直接查看 0000011 00B =00CH 处的页表项，有效位为 1，取出物理页号 19H=11001B，和页内偏移 111 1010B 拼接成物理地址：11001 111 1010B。根据中间 4 位 1110 直接找到 cache 第 14 行(即：第 E 行)，有效位为 1，且标记为 33H=110011B，正好等于物理地址高 6 位，故命中。根据物理地址最低两位 10，取出字节 2 中的内容 4AH=01001010B。

七、以下三个 I/O 程序 A、B、C 的输出各自是什么（5 分）?

```
#include <stdio.h>
A:  int main()
    {
        fprintf(stdout, “hello”);
        fprintf(stderr,”world!”);
        return 0;
    }
B:  int main()
    {
```

```
        fprintf(stdout, “hello”);
        fprintf(stderr,”world!\n”);
        return 0;
    }
C:  int main()
    {
        fprintf(stdout, “hello\n”);
        fprintf(stderr,”world! ”);
        return 0;
    }
```

答案:

A: world!hello

B: world!
   hello

C: hello
  World!