第 3 次作业

1. 在x86-32 CPU中，逻辑地址由哪两部分组成？每个段与段寄存器之间有何对应的要求？

答案：逻辑地址由段地址和偏移地址组成。x86-32 CPU 的程序由代码段、堆栈段、数据段和附加数据段，它们与段寄存器的对应关系如下：
CS：给出当前代码段首地址（取指令指针为 IP/EIP）。
SS：给出当前堆栈段首地址（堆栈指针为 SP/ESP）。
DS：给出当前数据段首地址。
ES、FS、GS：给出当前附加数据段首地址。
Intel CPU 有哪六种寻址方式？各种寻址方式中，操作数的地址如何得到？

2. Intel CPU 有哪六种寻址方式？各种寻址方式中，操作数的地址如何得到？

答案：
寄存器寻址: R，操作数在 CPU 内部的寄存器 R 中，即为（R）。
立即寻址: n，操作数 n 作为指令编码的一部分，存贮在主存贮器中。
直接寻址: 段寄存器:[n] 或 变量+ n，操作数在主存贮器中。偏移地址为 n。
寄存器间接寻址: [R]，操作数在主存贮器中。偏移地址为寄存器 R 的内容，段地址在 DS 或 SS 中。
变址寻址[R*F+V]，操作数在主存贮器中。偏移地址= (R)*F+V，段地址在 DS 或 SS 中。
基址加变址[BR+IR*F+V]，操作数在主存贮器中。偏移地址= (BR) +(IR)*F+V，段地址在 DS 或 SS 中。

3. 保护方式的段寄存器的内容表示什么？它与实方式的段寄存器的内容有什么不同？

答案：在保护方式下，段寄存器的内容不再指示段的起始地址，其高 13 位存放的是一个索引值，根据这个索引值，可以找到这个段的描述符（4 字节），从段的描述符中可以得到需要访问的存贮单元的物理地址。实方式的段寄存器存放的是段的开始地址，保护方式的段寄存器保存的是一个索引值（不是段的开始地址）。

4. 在保护方式下，简述 CPU 如何获得用户程序 A 的内存单元 ES:[12345H]的物理地址。

答案：

(1) 从 GDTR 寄存器中获取 GDT（全局描述符表段）的地址；
(2) 在 GDT 表中，以 LDTR 的高 13 位作为索引，取出程序 A 的 LDT（局部描述符表）段的描述符；
(3) 根据 A 的 LDT 段的描述符找到程序 A 的 LDT 段（LDT_A）；
(4) 用 ES 的高 13 位，作为索引，在 LDT_A 段中找到 ES 段的描述符 P_A；
(5) 从描述符 P_A 中获得段（ES）的基址、并与偏移地址 12345H 一起，形成最终的物理地址。

5. 在实模式下，编写一个程序段，将物理地址为 12345H 的长字单元（4 字节）的内容拷贝到 EAX 寄存器中。

答案：

```
MOV   AX, 1234 H
MOV   DS, AX
MOV   EAX, DS:[5]
```

6. 实方式下段内的第一个字节的偏移地址是多少？为什么实方式下每个段的起始物理地址一定能被 16 整除？

答案：实方式的物理地址是：段寄存器的内容*16＋ 偏移地址，而一个段的第一个存贮单元的偏移地址是 0，所以段的第一个存贮单元的物理地址是：段寄存器的内容*16＋0。因此，每个段的起始物理地址（该段的第一个存贮单元的物理地址）一定能被 16 整除。

7. 在实模式下，分析下面的结果。

右边是一段内存分布图
执行下面2条语句后，
(EAX)=?，(EBX)=?

```
MOV  AX, 8[BX][SI]
MOV  EBX, -8[EDI*2][EBP]
```

执行前：(DS) = 1234H
(ES) = 1200H
(SS) = 1200H
(EAX) = 89ABCDEFH
(EBX) =10H
(ESI) = 20H
(EBP) = 300H
(EDI) = 40H

物理地址	
12375H	02 H
12376H	05 H
12377H	1E H
12378H	28 H
12379H	32 H
1237AH	12 H
1237BH	34 H
1237CH	52H

答案：

(1) MOV AX, 8[BX][SI] ⇔ MOV AX, DS:[BX+SI+8] ⇔
MOV AX, DS:[38] ； 从物理地址 12340+38=12378 处取 2 字节
（AX）= 3228 H

(2) MOV EBX, -8[EDI*2][EBP] ⇔ MOV EBX, SS:[EBP+2*EDI-8] ⇔

MOV EBX, SS:[300+2*40-8] ;　物理地址 12000+378=12378

（EBX）= 34 12 32 28 H

8. C 程序中变量访问对应的寻址方式

```
int    g;
void local_variable_visit()
{
      int    a[5];
      int    i, *p;
      g = 1;
      a[0] = 10;
      a[1] = 20;
      i = 2;
      a[i] = 30;
      *(a + 3) = 40;
      p = &a[0];
      p += 4;
      *p = 50;
      printf("%d %d %d %d %d \n", g, a[0], a[1], a[2], a[3], a[4]);
}
```

观察反汇编代码，写出 printf 语句之前，给 g、a[0]、a[1]、a[2]、a[3]、a[4]赋值的汇编语句（赋值前进行的地址计算中间过程语句不需要写出来），指出它们分别使用的是什么寻址方式。指出 g 和 a 的地址表达形式。

答案：

g = 1;

006E4EC2　mov　　dword ptr ds:[006EA13Ch], 1　　//直接寻址

a[0] = 10;

006E4ED4　mov　　dword ptr [ebp+ecx-1Ch], 0Ah　　//基址加变址寻址

a[1] = 20;

006E4EE4　mov　　dword ptr [ebp+eax-1Ch], 14h　　//基址加变址寻址

a[i] = 30;

006E4EF6　mov　　dword ptr [ebp+eax*4-1Ch], 1Eh　　//基址加变址寻址

*(a + 3) = 40;

006E4EFE　mov　　dword ptr [ebp-10h], 28h　　//变址寻址

*p = 50;

006E4F20　mov　　dword ptr [eax], 32h　　//寄存器间接寻址

g 的地址：ds:[006EA13Ch]

由于 a+3 的地址为[ebp-10h]，所以 a 的地址为：[ebp-10h-3*4] = [ebp-1Ch]

9. 简述指令指示器EIP的作用。

答案： EIP 是指令指示器，保存下一条将被执行的指令的偏移地址，其值由 CPU 硬件自动设置。

10. 简述 x86-32 CPU 执行一条指令的过程（从取指机器码开始，直到指令执行完毕）。

答案：

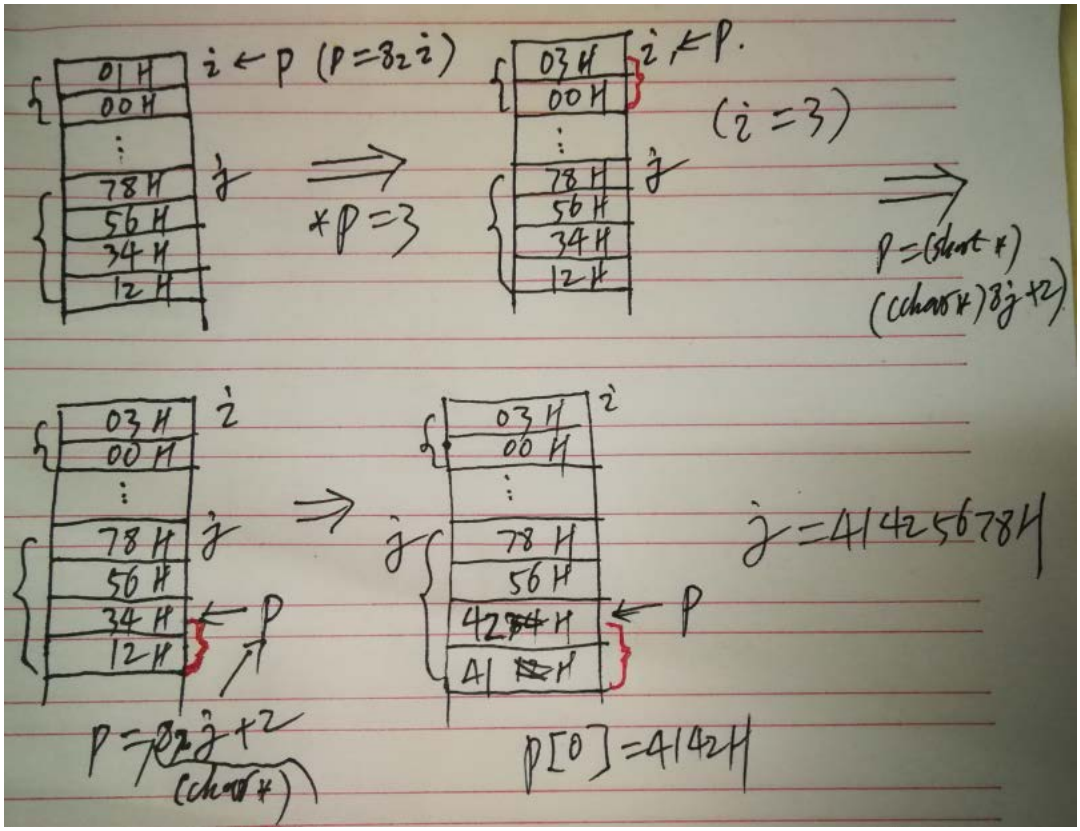
(1) CPU 的取指部件从存贮器的地址 CS:EIP 处取出一条指令的机器码到 CPU；
(2) CPU 的指令译码部件对指令进行分析；
(3) CPU 的执行部件根据译码部件的分析结果执行指令；
(4) CPU 将执行结果保存到相应的地方（CPU、存贮器、IO 端口）。

11. 分析下面的 C 程序的运行结果，需要画出相关变量存贮图的变化过程。

(1) 程序 1

```
#include <stdio.h>
int main()
{
    short int   i = 1;
    long int j = 0x12345678;
    short int *p = &i;
    *p = 3;
    printf("%d", i);
    p = (short int *)( (char *)&j + 2 );
    p[0] = 0x4142;
    printf("%x ", j);
}
```

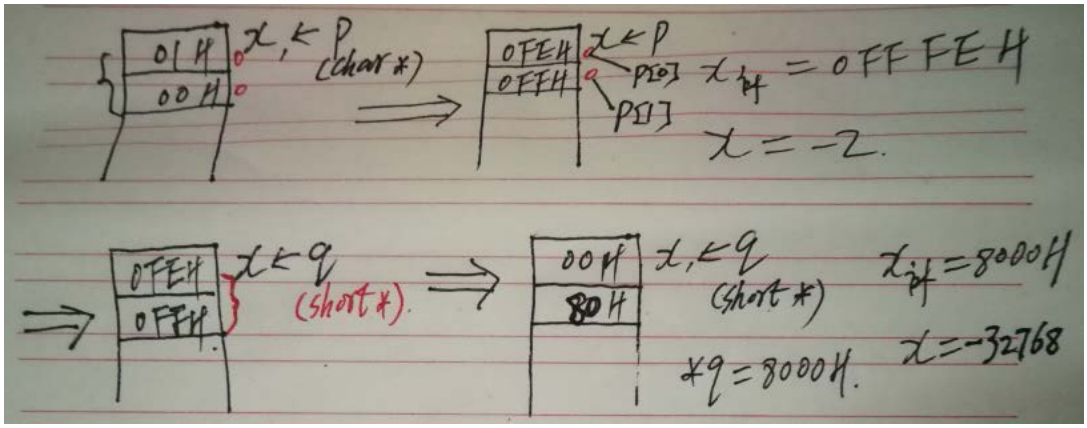
答案： 3 41425678



(2) 程序 2

```
#include <stdio.h>
void main()
{
	short int x = 1;
	unsigned char *p;
	p = (unsigned char *)&x;
	p[0] = 0xFE;
	p[1] = 0xFF;
	printf(“%d”, x);
	unsigned short *q = &x;
	*q = 0x8000;
	printf(“%d”, x);
}
```

答案： **-2　-32768**



(3) 程序 3

```
#include <stdio.h>
union {  long    num;
          struct {  unsigned short n1;
                    unsigned short n2;
                  } b;
        } a;
void main()
{
      a.num = 0x12345678;
      printf("%x    %x \n", a.b.n1, a.b.n2);
      a.num *= -1;
      printf("%x    %x \n", a.b.n1, a.b.n2);
}
```

答案： **5678　1234　A988　EDCB**