3.5

(1) 偏移地址=D=5119H

(2) 偏移地址=(BX)+D=6319H

(3) (BX)=1200H,

相对地址为(224AU)=0600H

偏移地址为1200H+0600H=1800H

3.6 ~~MOV BX,[2000H]~~

~~MOV AX,[2000H+2]~~

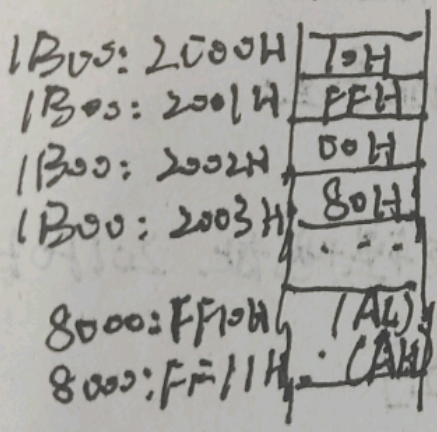
~~MOV ES,AX~~

~~MOV AX ES:[BX]~~

MOV BX,2000H

LES DI,[BX]

MOV AX,ES:DI



3.7 (1) OBJ=0624H+02H+27H=064DH

(2) OBJ=0624H+02H+6BH=0691H

(3) OBJ=0624H+02H+3AH=05ECH

3.8

(1) 立即寻址，无物理地址

(2) 寄存器寻址，(BX)=0100H为操作数无物理地址

(3) 直接寻址，物理地址为20100H

(4) 直接寻址，物理地址为20050H

(5) 寄存器间接寻址，物理地址：20100H

(6) 寄存器间接寻址，物理地址为：21100H

(7) 寄存器间接寻址，物理地址为150H

(8) 寄存器间接寻址，物理地址：200A0H

(9) 寄存器相对寻址，物理地址：20110H

(10) 寄存器相对寻址，物理地址：20150H

(11) 基址变址寻址，

物理地址：201A0H

(12) 相对基址变址寻址，物理地址 201F0H

3.9

(1)
```
MOV AX, [BX+SI+2]
MOV [BX+14], AX
```

(2)
```
MOV AX, ARRAY[BX]
MOV ARRAY[BX+2], AX
```

3.10.

MOV AX, TABLE 将 TABLE 中的数据存入 AX 中，

LEA AX, TABLE，将 TABLE 的有效地址传入 AX.

题目 1

1. 一致性语义

执行 REP MOVS 时，处理器会自动处理源地址和目标地址的递增（或递减），并且可能会对内存访问进行一些优化。REP MOVS 不保证内存的强一致性，特别是在多核处理器中，其他处理器可能在操作完成之前读取到未更新的数据。

memcpy 可以在实现中包含内存屏障，确保在多线程环境下的内存一致性。具体实现可能依赖于编译器和系统的内存模型。

memcpy 的实现可以更灵活，能够处理不同平台的特定优化，包括使用 SIMD 指令集等。

2. 性能不同

REP MOVS 可能在执行时不充分利用 CPU 的缓存机制，导致更多的内存访问和缓存未命中。

memcpy 的实现可能会包含优化，以减少内存访问次数，例如使用缓存行的对齐处理。

题目 2

ERMSB 允许处理器在执行 REP MOVS 时，能够有效地将内存复制操作提升为更高效的内存块操作。这意味着处理器可以在硬件层面上优化内存传输，减少每次内存访问的开销。

ERMSB 通过减少 CPU 对内存的访问次数，提高了内存复制的效率。它能够在内部处理更大的数据块，减少了对主内存的频繁访问，能够更好地利用 CPU 的缓存机制，减少缓存未命中的情况，这样可以进一步提升性能。通过在缓存中处理数据，ERMSB 可以避免反复访问主内存。

题目 3

1. 对 LODS 增加 REP 前缀有意义。通过增加 REP 前缀，LODS 可以重复执行多次，依次从内存中加载多个数据项到寄存器。简化手动循环的实现，避免编写复杂的循环逻辑。
2. 对 STOS 增加 REP 前缀有意义。可以高效地将寄存器中的数据重复存储到内存中的多个位置。这在需要初始化数组或数据结构时特别有用。

题目 4

溢出发生在有符号数的运算中，表示结果超出了可以表示的范围。若两个数同号相加，结果与操作数符号不同，则发生溢出。

进位发生在无符号数的运算中，表示结果超出了可以表示的最大值。

INTO 指令确保有符号数运算的安全性，溢出通常表示更严重的错误，尤其是在有符号数运算中，可能导致数据解释错误。中断处理相对耗时，进位的处理通常可以在程序逻辑中轻松实现，不需要中断的开销。