**一、数据传送指令**

1. **MOV（Move）**
   * **作用**：将源操作数（立即数、寄存器、内存单元）传送到目标操作数（寄存器、内存单元）。
   * **用法**：MOV 目标, 源，如 MOV AX, 10（将 10 送 AX）。
2. **LEA（Load Effective Address）**
   * **作用**：将内存操作数的有效地址加载到寄存器。
   * **用法**：LEA 寄存器, 内存地址，如 LEA AX, [BX+5]（取 BX+5 的地址给 AX）。
3. **XCHG（Exchange）**
   * **作用**：交换两个操作数（寄存器之间、寄存器与内存间）的内容。
   * **用法**：XCHG 操作数1, 操作数2，如 XCHG AX, BX（交换 AX 和 BX）。
4. **XLAT（Translate）**
   * **作用**：根据 AL 索引，从 BX 指向的表中取对应字节到 AL。
   * **用法**：先设表地址到 BX，XLAT（隐含使用 AL 为索引）。
5. **PUSH（Push onto Stack）**
   * **作用**：将操作数（寄存器、内存单元）压入堆栈。
   * **用法**：PUSH 操作数，如 PUSH AX（AX 压栈）。
6. **POP（Pop from Stack）**
   * **作用**：从堆栈弹出数据到目标操作数。
   * **用法**：POP 目标，如 POP BX（栈顶数据送 BX）。
7. **PUSHF / POPF**
   * **作用**：分别将标志寄存器压栈、弹出。
   * **用法**：PUSHF（标志寄存器压栈）；POPF（标志寄存器弹出）。
8. **LDS（Load DS with pointer）**
   * **作用**：将源操作数的段地址和偏移量送 DS 和指定寄存器。
   * **用法**：LDS 寄存器, 源，如 LDS SI, [DI]（[DI] 地址送 SI，段地址送 DS）。
9. **LES（Load ES with pointer）**
   * **作用**：类似 LDS，但段地址送 ES。
   * **用法**：LES 寄存器, 源。
10. **LAHF（Load AH with Flags）**
    * **作用**：将标志寄存器低 8 位（SF、ZF、AF、PF、CF）送 AH。
    * **用法**：LAHF。
11. **SAHF（Store AH into Flags）**
    * **作用**：将 AH 内容送标志寄存器低 8 位。
    * **用法**：SAHF。
12. **IN（Input）**
    * **作用**：从端口读数据到累加器（AL/AX）。
    * **用法**：IN 累加器, 端口号，如 IN AL, 20H（从端口 20H 读数据到 AL）。
13. **OUT（Output）**
    * **作用**：将累加器（AL/AX）数据输出到端口。
    * **用法**：OUT 端口号, 累加器，如 OUT 20H, AL（AL 数据送端口 20H）。

**二、算术运算指令**

1. **ADD（Add）**
   * **作用**：两数相加，结果存目标操作数。
   * **用法**：ADD 目标, 源，如 ADD AX, BX（AX + BX → AX）。
2. **ADC（Add with Carry）**
   * **作用**：带进位加法，用于多字节运算。
   * **用法**：ADC 目标, 源，如 ADC AX, BX（AX + BX + CF → AX）。
3. **AAA（Ascii Adjust After Addition）**
   * **作用**：对非压缩 BCD 码加法结果调整为 ASCII 码。
   * **用法**：紧跟加法指令后，如 ADD AL, BL；AAA。
4. **DAA（Decimal Adjust After Addition）**
   * **作用**：对压缩 BCD 码加法结果进行十进制调整。
   * **用法**：紧跟加法指令后。
5. **SUB（Subtract）**
   * **作用**：两数相减，结果存目标操作数。
   * **用法**：SUB 目标, 源，如 SUB AX, BX（AX - BX → AX）。
6. **SBB（Subtract with Borrow）**
   * **作用**：带借位减法，用于多字节运算。
   * **用法**：SBB 目标, 源，如 SBB AX, BX（AX - BX - CF → AX）。
7. **AAS（Ascii Adjust After Subtraction）**
   * **作用**：对非压缩 BCD 码减法结果调整为 ASCII 码。
   * **用法**：紧跟减法指令后。
8. **DAS（Decimal Adjust After Subtraction）**
   * **作用**：对压缩 BCD 码减法结果进行十进制调整。
   * **用法**：紧跟减法指令后。
9. **MUL（Multiply）**
   * **作用**：无符号数乘法。单字节时 AL \* 源 → AX；字时 AX \* 源 → DX:AX。
   * **用法**：MUL 源，如 MUL BL（AL \* BL → AX）。
10. **IMUL（Integer Multiply）**
    * **作用**：有符号数乘法，规则同 MUL。
    * **用法**：IMUL 源。
11. **AAM（Ascii Adjust After Multiply）**
    * **作用**：对非压缩 BCD 码乘法结果调整为 ASCII 码。
    * **用法**：紧跟乘法指令后。
12. **CBW（Convert Byte to Word）**
    * **作用**：将 AL 中字节符号扩展到 AX（如 AL < 80H，则 AX = 00ALH；否则 AX = FFALH）。
    * **用法**：CBW。
13. **DIV（Divide）**
    * **作用**：无符号数除法。单字节时 AX ÷ 源，商 → AL，余数 → AH；字时 DX:AX ÷ 源，商 → AX，余数 → DX。
    * **用法**：DIV 源，如 DIV BL（AX ÷ BL）。
14. **IDIV（Integer Divide）**
    * **作用**：有符号数除法，规则同 DIV。
    * **用法**：IDIV 源。
15. **AAD（Ascii Adjust Before Division）**
    * **作用**：对非压缩 BCD 码除法前调整为二进制数。
    * **用法**：紧跟除法指令前。
16. **CWD（Convert Word to Doubleword）**
    * **作用**：将 AX 中字符号扩展到 DX:AX（如 AX < 8000H，则 DX = 0；否则 DX = FFFFH）。
    * **用法**：CWD。
17. **CMP（Compare）**
    * **作用**：比较两数（执行减法但不存结果，仅改标志位）。
    * **用法**：CMP 目标, 源，如 CMP AX, BX（影响 ZF、SF、CF 等标志）。
18. **INC（Increment）**
    * **作用**：操作数加 1。
    * **用法**：INC 操作数，如 INC AX（AX + 1 → AX）。
19. **DEC（Decrement）**
    * **作用**：操作数减 1。
    * **用法**：DEC 操作数，如 DEC BX（BX - 1 → BX）。
20. **NEG（Negate）**
    * **作用**：操作数求补（按位取反 + 1）。
    * **用法**：NEG 操作数，如 NEG AX（AX 内容取补）。

**三、逻辑运算指令**

1. **AND（Logical And）**
   * **作用**：两操作数按位逻辑与（全 1 则 1，否则 0）。
   * **用法**：AND 目标, 源，如 AND AX, 0FH（保留 AX 低 4 位，其余清 0）。
2. **OR（Logical Or）**
   * **作用**：两操作数按位逻辑或（有 1 则 1，全 0 则 0）。
   * **用法**：OR 目标, 源，如 OR AX, 0FH（置位 AX 低 4 位）。
3. **NOT（Logical Not）**
   * **作用**：操作数按JPL位取反（1 变 0，0 变 1）。
   * **用法**：NOT 操作数，如 NOT AX（AX 内容取反）。
4. **XOR（Exclusive Or）**
   * **作用**：两操作数按位异或（相同为 0，不同为 1）。
   * **用法**：XOR 目标, 源，如 XOR AX, AX（AX 清零）。
5. **TEST**
   * **作用**：对两操作数逻辑与，但不存结果，仅改标志位（常用于检测位状态）。
   * **用法**：TEST 目标, 源，如 TEST AX, 80H（检测 AX 最高位是否为 1，不改变 AX）

**4）位移指令**

* **SHL（Shift Logical Left）**：逻辑左移，最高位进 CF，低位补 0。
  + 例：SHL AX, 1（AX 左移 1 位）；SHL BX, CL（BX 左移 CL 位）。
* **SHR（Shift Logical Right）**：逻辑右移，最低位进 CF，高位补 0。
  + 例：SHR DX, 1（DX 右移 1 位）。
* **SAL（Shift Arithmetic Left）**：算术左移，功能同 SHL。
  + 例：SAL CX, CL（CX 左移 CL 位）。
* **SAR（Shift Arithmetic Right）**：算术右移，保持符号位，最低位进 CF。
  + 例：SAR AX, 1（AX 右移 1 位，符号位不变）。
* **ROL（Rotate Left）**：循环左移，最高位同时进 CF 和最低位。
  + 例：ROL BL, 1（BL 循环左移 1 位）。
* **ROR（Rotate Right）**：循环右移，最低位同时进 CF 和最高位。
  + 例：ROR AX, CL（AX 循环右移 CL 位）。
* **RCL（Rotate Through Carry Left）**：带进位循环左移，CF 参与循环。
  + 例：RCL DH, 1（DH 带进位左移 1 位）。
* **RCR（Rotate Through Carry Right）**：带进位循环右移，CF 参与循环。
  + 例：RCR CX, CL（CX 带进位右移 CL 位）。

**5）串操作指令**

* **MOVS（Move String）**：将源串（DS:SI）数据移至目的串（ES:DI）。
  + 例：MOVS BYTE PTR [DI], [SI]（字节串移动）。
* **CMPS（Compare String）**：比较源串与目的串（DS:SI 与 ES:DI）。
  + 例：CMPS WORD PTR [DI], [SI]（字串比较）。
* **SCAS（Scan String）**：用 AL/AX 扫描目的串（ES:DI）。
  + 例：SCAS BYTE PTR [DI]（字节扫描）。
* **LODS（Load String）**：将源串（DS:SI）数据载入 AL/AX。
  + 例：LODS WORD PTR [SI]（字串载入）。
* **STOS（Store String）**：将 AL/AX 数据存入目的串（ES:DI）。
  + 例：STOS BYTE PTR [DI]（字节存储）。
* **重复前缀**：
  + **REP（Repeat）**：重复串操作，直到 CX = 0。
  + **REPE / REPZ（Repeat While Equal / Zero）**：相等（ZF=1）且 CX ≠ 0 时重复。
  + **REPNE / REPNZ（Repeat While Not Equal / Not Zero）**：不相等（ZF=0）且 CX ≠ 0 时重复。

**6）控制转移指令**

* **JMP（Jump）**：无条件跳转。
  + 例：JMP LABEL（跳转到 LABEL 处）。
* **Jxx（Conditional Jump）**：条件跳转（如 JE 等于跳转，JNE 不等跳转）。
  + 例：JE TARGET（ZF=1 时跳转）。
* **CALL（Call Subroutine）**：调用子程序。
  + 例：CALL PROC\_NAME（调用 PROC\_NAME 子程序）。
* **RET（Return）**：从子程序返回，RET n 中 n 用于调整堆栈。
  + 例：RET（无参数返回）；RET 4（返回并调整堆栈）。
* **INT（Interrupt）**：触发中断。
  + 例：INT 21H（调用 DOS 中断 21H）。
* **IRET（Interrupt Return）**：从中断返回。

**7）处理器控制指令**

* **CLC（Clear Carry Flag）**：清进位标志（CF=0）。
* **STC（Set Carry Flag）**：置进位标志（CF=1）。
* **CLD（Clear Direction Flag）**：清方向标志（DF=0，串操作递增）。
* **STD（Set Direction Flag）**：置方向标志（DF=1，串操作递减）。
* **CLI（Clear Interrupt Flag）**：关中断（IF=0）。
* **STI（Set Interrupt Flag）**：开中断（IF=1）。
* **CMC（Complement Carry Flag）**：进位标志取反（CF 取反）。
* **NOP（No Operation）**：空操作，不执行实际功能。
* **HLT（Halt）**：暂停处理器，直到中断或复位。