ASM Cheatsheet

wacky6 / Jiewei Qian (C) CC-BY-NC-SA-3.0

I hope this document can help you pass the ~~not so~~ difficult 微机接口 exam. However, you MUST NOT hold me accountable for failing it, or for mistakes in this document.

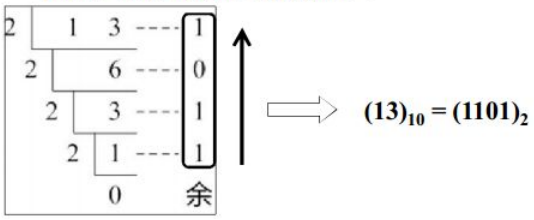
:D

Acknowledgment:
This document contains screenshots of Prof. Ni Xiaojun's slide for Assembly Language introduction, part of Computer Interface course. (NJUPT-CS).

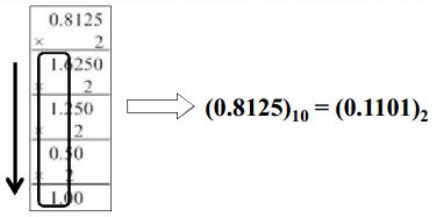
1*. 数制转换

（1） 10 -> 2

➢对于十进制整数，转换算法：除2取余，直到商为零为止，倒排。例如，10进制数13：



➢对于十进制小数，转换算法：乘2取整，直到乘积的整数部分为零为止，顺排。例如，10进制小数0.8125：



** 整数：短除取余反序； 小数：连乘直至为整数，依次取整数位

** 整数、小数 分开计算

** 也可 10->16 再将各位转 2

（2） 2->8, 2->16, 8->2, 16->2

每 3/4 个二进制位对应一个 8/16 进制位

整数位数不足在**左**侧添 0 补齐，小数位数不足在**右**侧添 0 补齐

（3） BCD

8421BCD， 4 位二进制，9 以上为非法码

2. ASCII 码

0～9的ASCII码为	30H ~ 39H
A～F的ASCII码为	41H ~ 46H
回车符的ASCII码为	0DH
换行符的ASCII码为	0AH

（1） 重要 ASCII 码

（2） DOS 中输出回车： 0x0D, 0x0A （先把光标移到行首，再把光标移到下一行）

3. 码制

（1） 原码：最高位为符号位，数值部分为数的绝对值

（2） 反码：最高位为符号位，数值部分：正数->绝对值，负数->绝对值按位取反

（3） 补码*：正数->与原码相同； 负数：保留符号位，数值位为**绝对值按位取反后加 1**

（4） 补码运算*：二进制数直接相加，最高位进位丢弃。

（5） 溢出*： 无符号数：C 标志（进位标志）为 1； 有符号数：O 标志（溢出标志）为 1

4. 8086

（1）8084 是 CISC（复杂指令集计算机）处理器
（2）实模式：（DOS 下）
　　可以访问 00000H-FFFFFH （1M 的内存）
　　只进行分段，不进行分页，逻辑段最大 64K，段寄存器存放段基址
（3）保护模式：（Win 下）
　　支持多任务，4G 内存，分段分页
（4）486 一共 32 根地址线，所以可访问 4G（2^32 Byte）内存
（5）数据为小端存储（Little Endian），最不重要的字节在前（Least Significant Byte）

5. 寄存器

通用寄存器： AX（Accumulator） BX（Base pointer） CX（Counter） DX（Data）
　　通用寄存器 16 位，分为高/低 8 位，用 nH, nL 方位
　　IP（Instruction）：指令指针， SP（Stack）：堆栈指针
　　BP（Base）：基址指针，SI（Source）：源变址，DI（Dest.）：目标变址
寻址程序： CS + IP （通常 CS 省略不写）
寻址数据： DS/ES + SI/DI/BP/BX
寻址堆栈： SS + SP/BP

6. 地址、寻址

（1）物理地址的形成： 段地址×(2^4) + 偏移地址
　　eg: 1000H : 2345H = (1000H << 4) + 2345H == 12345H
（2）段大小：最大 64K（0000H - FFFFH = 64K），最小 16Byte（0H - FH =16Bytes）
（3）

逻辑段	段基址	偏移地址
代码段	CS	IP
堆栈段	SS	SP
数据段	DS	根据不同的寻址方式选择BX/SI/DI
附加段	ES/FS/GS	

（4）DS、ES 的初值需要手动设置

```
ASSUME CS:CODE, DS:DATA
MOV AX, DATA
MOV DS, AX  /  MOV ES, AX
```

7. 指令系统

（1）指令长度：1-16 字节； 操作码 + 操作数
（2）ASM 写法：

```
标号：  操作符    操作数    ;注释
```

8*. 寻址方式

三类：（三类七种， 立即寻址、寄存器寻址各算一种）

（1） 立即寻址 （MOV AX, 00H） 立即数

（2） 寄存器寻址 （MOV AX, 00H） 寄存器

（3） 存储器寻址 （MOV AX, [SI]） 存储区（有[]），包括 5 种方式

存储器寻址的五种方式（仅 16 位）

（1） 直接寻址： （MOV AX, DS:[1234H]） []内是一个数值

（2） 寄存器间接寻址： （MOV AX, DS:[SI]） []内是一个(BX/SI/DI/BP)

（3） 基址寻址： （MOV AX, DS:[BP+4]） []内是**(BP/BX)** + 偏移量

（4） 变址寻址： （MOV AX, DS:[SI+12]） []内是**(SI/DI)** + 偏移量

（MOV AX, [8*ESI+5]） []32 位可以用比例因子

注：16 位模式下，变址模式不允许使用比例因子

比例因子必须是 1,2,4,8

（5） 基址加变址： （MOV AX, [BX+SI+5]） []内包括 BP/BX + SI/DI + 偏移量

段约定：

（1） BP：默认 SS（堆栈），否则，显示说明（如 DS:[BP]）

（2） BX, SI, DI, EAX-EDX, ESI, EDI：默认 DS（数据），否则，显示说明（如 ES:[SI]）

9*. 标志寄存器

- 进位标志C（Carry Flag）—— 当运算结果的最高位产生一个进位或借位，则C=1，否则C=0；
- 溢出标志O（Overflow Flag）—— 在算术运算中，有符号数的运算结果超出了8位或16位有符号数能表达的范围，则O=1，否则O=0；
- 符号标志S（Sign Flag）—— 结果的最高位（D_{15}或D_7）为1，则S=1，否则 S=0；
- 零标志Z（Zero Flag）—— 若运算的结果为0，则Z=1，否则Z=0；
- 奇偶标志P（Parity Flag）—— 若结果中‘1’的个数为偶数，则P=1，否则，P=0；
- 辅助进位标志A（Auxiliary Flag）—— 在运算过程中，由低半字节（第3位）向高半字节有进位或借位，则A=1，否则A=0。

（1）给两个数运算，问 C,O,S,Z,P,A 状态

（2）O 标志：两数运算结果的符号位与源数符号位均不同

eg: 01111111B + 00000001B = 10000000B (O 标志为 1)

（3）标志位被用来测试转移：

J(C/O/S/Z/P/A)为对应标志 1 时转移

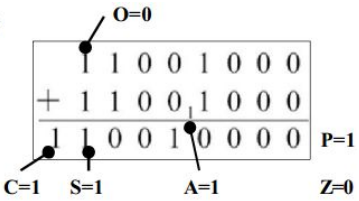
JN(C/O/S/Z/P/A)为对应标志为 0 时转移

常与 CMP A,B 用： JZ/JNZ = JE/JNE（A=B），JC/JNC=JB,JAE / JA,JBE（A<B)

（4）

例：下列指令执行后 AL=?
A.C.O.P.S.Z 为何值?

```
MOV AL,0C8H
ADD AL,0C8H
```



10. 伪指令

（1） DB, DW, DD （变量定义， Define Byte/Word/DWord， 1,2,4 字节）

小端存储，内存数据与定义的数字节顺序**相反** （NUM DW, 1234H --内存-> 34H 12H）

（2） N DUP(?) 表示 N 个随机数

（3） N DUP(‘A’) 表示 N 个’A’的 ASCII 码

（4） N DUP(0) 表示 N 个 0

（5） V EQU 123 定义常量 V 为 123，相当于#define V (3)，代码不会有 V 的存储区

（6） $ 一般用来统计字符串长度，表示汇编计数器的值

```
BUF DW “doge”
LEN EQU $-BUF   ; LEN=4
```

（7） SEG DATA 定义名为 DATA 的段

注：**不是定义数据段**，数据段是通过 ASSUME DS:DATA, MOV DS,DATA 指定的段名随意，段的功能是段寄存器指定的

（8） OFFSET VAR 取 VAR 的偏移地址

（9） [TYPE] PTR 按[TYPE]取 PTR 地址，TYPE 为 BYTE,WORD,DWORD；用于类型转换

11. 8086 基础指令

（1） MOV DEST, SRC 传送指令

把 SRC 移动到 DEST 中， SRC、DEST 为同类型操作数（或可推导操作数长度）

DEST 不能是立即数

不能把立即数直接送到段寄存器（不能 MOV DS, 3000H）

不能在存储单元/段间传送数据 （不能 MOV [2000H], [1000H]），必须寄存器中转

注意立即数移到内存中按照小段存储，字节序反序

（2） LEA DEST, SRC 取偏移地址

等价于 MOV DEST, OFFSET SRC

（3） △ XCHG OPR1, OPR2 交换 OPR1, OPR2

寄存器、寄存器 或 寄存器、存储器 之间交换数据，不能使用段寄存器

（4） PUSH VAR 把 VAR 压入堆栈

（5） POP VAR 从堆栈取出 VAR

PUSH, POP 配对使用

不能确定操作数类型时，显示说明

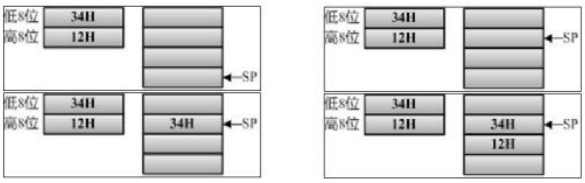
eg: PUSH WORD PTR [BX]

对于16位数据

SP-2 → SP

数据的低8位 → SS:[SP]

数据的高8位 → SS:[SP-1]



（6） △ LAHF FLAG 低 8 位送到 AH

（7） △ SAHF AH 送到 FLAG 低 8 位

（8） PUSHF/POPF 把 FLAG 进栈/出栈

（9） ADD/SUB DEST, SRC 加减法, DEST = DEST (+/-) SRC

（10） △ ADC/SBB 带进位/借位的加减法， DEST = DEST (+/-) SRC (+/-) C-Flag

（11） INC/DEC DEST 加 1/减 1

（12） CMP A, B 按 SUB A,B 设置 FLAG，A,B 的值不变

（13） NEG DEST 求 DEST 的补码

（14） △ MUL/IMUL, DIV/IDIV SRC 乘法，除法

MUL SRC —— 无符号数乘法
源操作数： 通用寄存器、存储器(不能是立即数)
目的操作数： EDX, EAX(隐含)
执行的操作： 字节操作 (AH,AL) ← (AL)*(SRC)
字操作 (DX,AX) ← (AX)*(SRC)
双字操作 (EDX,EAX) ← (EAX)*(SRC)
注：该指令影响标志位C和O

注意：若结果的高半部分（字节相乘为AH，字相乘为DX）
为0 则 C=0，O=0
不为0 则 C=1，O=1

DIV SRC —— 无符号数除法
IDIV SRC —— 有符号数除法
源操作数： 通用寄存器、存储器（不能是立即数）
目的操作数： EDX, EAX（隐含）
执行的操作（具体进行何种操作由SRC的类型决定）：
字节操作 (AL) ← (AX) / (SRC) —— 商
(AH) ← (AX) / (SRC) —— 余数
字操作 (AX) ← (DX,AX) / (SRC) —— 商
(DX) ← (DX,AX) / (SRC) —— 余数
双字操作 (EAX) ← (EDX,EAX) / (SRC) —— 商
(DEX) ← (EDX,EAX) / (SRC) —— 余数
注：该指令对各标志位均无影响。

（15） △ MOVZX DEST, SRC　0 扩展传送，DEST 长度大于 SRC，用 0 填充空余位数

（16） △

加法： DAA　压缩的BCD码加法十进制调整指令；
　　AAA　非压缩的BCD码加法十进制调整指令；
减法： DAS　压缩的BCD码减法十进制调整指令；
　　AAS　非压缩的BCD码减法十进制调整指令；
乘法： AAM　非压缩的BCD码乘法十进制调整指令；
除法： AAD　非压缩的BCD码除法十进制调整指令。

（17） 传送类指令

```
JMP   OPRD            ;无条件转移
LOOP  OPRD            ; CX = CX-1, 如果 CX!=0, JMP OPRD, 否则执行下一条语句
LOOPE/LOOPZ OPRD      ; CX = CX-1, 如果 CX=0, JMP OPRD
CALL  SUBP            ; 调用子程序
RET                   ; 子程序返回
INT   n               ; 调用软中断
```

元语：

Above: 无符号大于　　Below: 无符号小于

Greater: 有符号小于　　Lesser: 有符号小于

Equal: 等于

Not:　非

元语组合成指令名：

J[N]{A/B/G/L}[E]　= JMP ON [Not] {Above, Below, Greater, Lesser} [Or Equal]

常见等价：

JB = JC/JNAE　　JNC = JNB/JAE

JZ = JE　　JZ = JNE

J[N]{Flag}　按符号位跳转，　见*标志寄存器*

（18） SHL/SAL/SHR/SAR　OPR, COUNT　逻辑/算术 左/右 移

（19） ROL/ROR/RCL/RCR　OPR, COUNT　循环左/右移，　**带 C 标志**循环左/右移

（20） AND/OR/XOR　DEST, SRC　按位与、或、异或

（21） TEST　DEST, SRC　按位与后改变标志位，不影响操作数

（22） NOT　OPR　按位非

（23） △ MOVS(B/W/D), STOS(B/W/D), LODS(B/W/D)　串操作

CX 表示串长度，使用 SI,DI 标识串位置，每次操作后 SI,DI 自动加 1

（24） 状态位设置

CL(Flag)　清除 Flag 位

ST(Flag)　设置 Flag 位

(Flag) = {C,D,I}

12. 宏指令

一个完整的源程序在结构上必须做到：
- 用方式选择伪指令说明执行该程序的微处理器类型；
- 用段定义语句定义每一个逻辑段；
- 用过程定义语句定义每一个子程序；
- 用ASSUME语句说明段约定；
- 用汇编结束语句说明源程序结束；
- 程序在完成预定功能之后，应能安全返回DOS。

（1） `.486` 方式选择

（2） `NAME SEGMENT USE16`
`NAME ENDS` ;注意 NAME 配对！
定义名字为 NAME 的段

（3） ASSUME 段约定
`ASSUME CS:CODE, DS:DATA`
仅仅是约定，并**没有设置段寄存器的值**，段寄存器的值需要在程序中用 MOV 设置

（4） `END BEGIN` 汇编结束标志
BEGIN 为程序入口点的标号

（5） `NAME PROC`
`NAME ENDP` ;注意 NAME 配对！
定义名字为 NAME 的过程

13. 程序设计

（1） COM，EXE 差别： COM 执行优先级高（优先执行 COM），COM 不分段

14. DOS 功能调用

```
MOV AH, 功能号
; 设置参数
INT 21H    ;注意 21H 不是 21
```

按功能号：

（1）`01H` 获取一个字符，**回显**，**相应** CTRL+C
AL = 字符的 ASCII 码， 如果 AL=0，输入时光标键，再次调用本功能获取扩展码

（2）`07H` 获取一个字符，**不回显**，**不响应** CTRL+C
参数同 01H 调用

（3）`08H` 获取一个字符，**不回显**，**相应** CTRL+C
参数同 01H 调用

（4）`02H` 显示一个字符
DL = 字符的 ASCII 码

（5）`09H` 显示`$-Terminated` 字符串
DS:DX = 字符串首地址
$不会被显示，光标会移动到字符串结束位置

（6）`0AH` 读取字符串到缓冲区
DS:DX = 缓冲区地址
缓冲区格式： 缓冲区长度 | 读入的长度 | ...字符串... | （|为字节分隔）
缓冲区长度为存储空间长度，读入字串不包括回车

15. BIOS 功能调用

```
MOV AH, 功能号
INT 中断号
```

（1） 字符读取： INT 16H

```
1. 00H      读取字符，不回显，相应 CTRL+C
     AL=ASCII 码，AL=0 则再次调用本功能获得扩展码
2. 01H      查询输入缓冲区
     Z-Flag=0，有输入；此时 AL=ASCII 码，AH=扩展码
     Z-Flag=1，没输入
```

（3）文本显示： INT 10H

```
1. 00H      设置屏幕方式
    设置 AL：
        AL 第 0 位：      0：黑白；  1：彩色
        AL 第 1 位：      0：40x25；  1：80x25
    执行后，屏幕清屏，光标回到左上角
2. 02H      设置光标位置
    设置 BH：页号
    DH,DL：光标行号、列号
3. 03H      获取光标位置
    设置 BH：页号
    CH,CL：光标顶部、底部扫描线的行号
    DH,DL：光标行号、列号
4. 13H      显示字符串
    AL:   字串格式
        AL 第 0 位:   0/1 显示后光标置于字串首/尾部
        AL 第 1 位:   0/1 字串不包含/包含属性字
                      如字串不包含属性字，属性字由 BL 设置
    BH:   显示页号
    BL:   属性字 AL=0,1 时有效
    CX:   串长度
    DH/DL: 字串起始位置的行号/列号    （与 02H,03H 一致）
    ES:BP: 字串首地址      （注意设置 ES!）
```

16. 宏、子程序编程

（1） 宏编程

```
NAME MACRO 参数表
    LOCAL  本地标号说明
    ; 宏体
ENDM        ;NAME 不配对，与 ENDS/ENDP 不同!
```

参数表传入的值会直接替换宏体中的参数名

（2） 子程序

```
NAME PROC
    ; 子程序体
NAME ENDP
```

参数传入可以用 BX, DX

返回值可以用 BX

17. 样例程序

```
    .486              ; 方式选择
DATA SEGMENT USE16    ; 数据段
DATA ENDS

CODE SEGMENT USE16    ; 代码段
    ASSUME CS:CODE, DS:DATA

BEG:  ; 主程序
    MOV AX, DATA
    MOV DS, AX

    ; Do something!

EXIT:                 ; 结束程序，返回 DOS
    MOV AH, 4CH
    MOV AL, 0
    INT 21H

CVTBL PROC
        PUSH BX
    ; Do something!
        POP  BX
        RET               ; 过程调用返回
CVTBL ENDP

CODE ENDS
        END BEG           ; 程序入口点
```

18. Glossary & Notes

（1）A-F 开头的十六进制数前面要加上 0　　（MOV AX, 0FFFFH）

（2）返回 DOS 调用：

```
NOV  AH, 4CH
MOV  AL, 返回值
INT  21H
```

（3）编译链接执行

```
TASM  NAME   ;编译
TLINK NAME   ;链接
NAME         ;执行
```

（4）MACRO 不要拼错了!!

This page is intentionally left blank.