東北大學
Northeastern University

汇编语言程序设计

主讲： 刘松冉

单位： 东北大学计算机学院
智慧系统国际联合实验室

联系方式： liusongran@cse.neu.edu.cn

个人主页： http://faculty.neu.edu.cn/liusongran
https://liusongran.github.io/

第四章 汇编语言

一. 汇编语句格式

二. 汇编语言中数据的表示方法

三. 运算符号

四. 伪指令

五. 汇编语言的上机过程

第四章 汇编语言

一. 汇编语句格式

1. 字符集
2. 汇编语句格式

一. 汇编语句格式

1. 字符集

- Microsoft的宏汇编语言由下列字符组成：

  1) 英文字母：A~Z 和 a~z

  2) 数字字符：0~9

  3) 算术运算符：+, –, *, /

  4) 关系运算符：<, =, >

  5) 分隔符：,, :, ;, (,), [,], (空格), TAB(制表符)

  6) 控制符：CR(回车), LF(换行), FF(换页)

  7) 其它字符： $, & , _(下划线), ., @, %, !

一. 汇编语句格式

2. 汇编语句格式

```
DSEG    SEGMENT                                  ;数据段开始
DATA1   DB      13H，26H                         ;原始数据
DATA2   DW      0                                ;保存结果单元
DSEG    ENDS                                     ;数据段结束
SSEG    SEGMENT         STACK                    ;堆栈段开始
SKTOP   DB      20 DUP(0)
SSEG    ENDS                                     ;堆栈段结束
CSEG    SEGMENT                                  ;代码段开始
        ASSUME          CS:CSEG,DS:DSEG, SS:SSEG
START：  MOV     AX,DSEG                         ;初始化数据段基址
        MOV     DS,AX
        MOV     AX,SSEG                          ;初始化堆栈段基址
        MOV     SS,AX
        MOV     SP,LENGTH SKTOP                  ;设段堆栈指针
        MOV     AL,DATA1                         ;取第一个数据
        ADD     AL,DATAI+1                       ;与第二个数据相加
        MOV     BYTE PTR DATA 2,AL               ;保存结果
        MOV     AH,4CH
        INT     21H                              ;返回DOS
CSEG    ENDS                                     ;代码段结束
        END     START                            ;源程序结束
```

汇编语句（每行程序代码）最多由四个域组成：

[标号]+操作符+操作数+[；注释]

2. 汇编语句格式 — 标号

- **定义**: 程序设计人员自己定义的符号, 用于代表内存单元的地址。
- **组成规则**：

1) 组成标号的字符：

A~Z, a~z, 0~9, ?, ., @, $, _(下划线)

2) 标号的最大长度为31(字符个数)

3) 标号的第一个字符不能是0~9的数字

4) 机器指令语句中标号必须以冒号结束

5) 伪指令语句中的标号不允许有冒号

2. 汇编语句格式 — 标号

- **标号的3个属性:**

  1) 标号所代表的段的值, 可用SEG算符得到

  2) 标号所代表的偏移量的值, 可用OFFSET算符得到

  3) 类型属性: 标号所代表的内存单元所存放的数据或指令代码的类型。共有7种：

标号类型	字节类型 (BYTE)	字类型 (WORD)	双字类型 (DWORD)	四字类型 (QWORD)	十字节类型 (TBYTE)	近类型 (NEAR)	远类型 (FAR)
类型值	1	2	4	8	10	-1	-2

2. 汇编语句格式

- **操作符**：操作符可以是指令助记符, 伪指令助记符，宏指令符号。
- **操作数**：操作数是操作符操作的对象，可以是数据本身,也可以是标号、寄存器名或算术表达式。
- **注释**：对指令功能的说明，目的实施自己或他人在阅读分析程序是方便。以“;”开始，以行终止符（换行）结束。

第四章 汇编语言

二. 汇编语言中数据的表示方法

1. 数据在机内的表示
2. 汇编语言中数据的书写形式

二. 汇编语言中数据的表示方法

1. 数据的机内表示

- **数据的符号**
  - 用0表示正，1表示负。对于定点表示有原码反码补码等表示方法。
- **数据的定点表示法**
  - 小数点在最低位之后,代表纯整数, 16位表示范围为–32768~+32767
  - 小数点在最高位和次高位之间，代表纯小数
    - 例如，对于二进制数01101000B，如果约定小数点在最低位之后，则表示68H (104Q)，如果在最高位和次高位之间，则为0.d (0.8125)。
  - 缺点：不灵活，运算过程数据无法表示（如10/3=3.333333…）

- **数据的浮点表示法**
  - 在浮点表示中, 分为尾数和阶码两部分。有多种表示方法，现在一般用IEEE的表示方法

二. 汇编语言中数据的表示方法

1. 数据的机内表示 – 浮点表示法

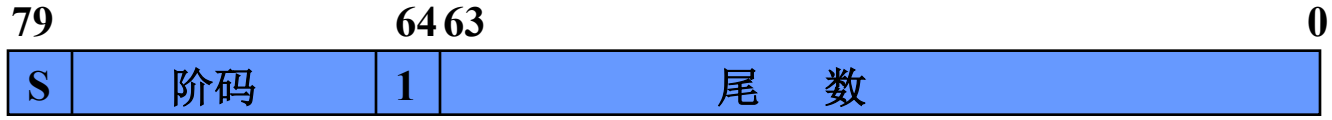
短实数
(4Bytes)

尾数23位, 阶码8位,过余量127(7FH)

长实数
(8Bytes)

尾数52位, 阶码11位,过余量1023(3FFH)

临时实数
(10Bytes)



尾数63位, 阶码15位,过余量16383(3FFFH)

1.fff…fff

二. 汇编语言中数据的表示方法

1. 数据的机内表示 – **浮点表示法**

短实数
(4Bytes)

尾数23位, 阶码8位,过余量127(7FH)

S　　：符号位，表示正负 0正数 1负数
阶码　：表示小数点在有效数字中的位置，过余码表示
尾数　：用于存放具体的数字

二. 汇编语言中数据的表示方法

1. 数据的机内表示 – 浮点表示法 （举例）

36.625Q

24.AH

转换为二进制为 100100.101

规格化后为 1.00100101×2^5

阶码为 101

加过余量127后 0111 1111+00000101=10000100

尾数(23位) 001 0010 1000 0000 0000 0000

符号&阶码&尾数 01000010 00010010 10000000 00000000

用16进制表示为 42 12 80 00

1.fff…fff

-36.625

符号位变化 11000010 00010010 10000000 00000000

C2 12 80 00

二. 汇编语言中数据的表示方法

1. 数据的机内表示 — 数据的表示范围

- **一个字节（字节数据）**
  - 无符号整数 0~255
  - 补码表示带符号整数 –128~127
- **二个字节（字数据）**
  - 无符号整数 0~655535
  - 补码表示带符号整数 –32768~+32767
- **四字节表示的浮点数据(规格化)**
  - 正数：$2^{127}\times(1-2^{-24}) \geq N \geq 2^{-127}\times 2^{-1}$
  - 负数：$2^{127}\times(-(1-2^{-24})) \leq N \leq 2^{-128}\times(-2^{-1})$
  - 零：阶码和尾数同时为0

二. 汇编语言中数据的表示方法

2. 汇编语言中数据的书写形式

- 二进制常数：0~1 以B结尾
- 八进制常数：0~7, 以O或Q结尾
- 十进制常数：0~9, 以D结尾或空
- 十六进制常数：0~9, A~F, 以H结尾
- 十进制科学记数法常数：以0~9，E和指数组成，如0.2467E+5
- 字符常数：用单引号或双引号给出，例如 ’a’，”国”
- 标识符：用符号定义伪指令定义的符号
- 表达式：用运算符号连接起来的以上各种数据

第四章 汇编语言

三. 运算符号

1. 数算术运算符
2. 逻辑运算符
3. 关系运算符
4. 分析算符
5. 组合算符
6. 字节分离算符
7. 记录专用算符
8. 算符的优先级

三. 运算符号

1. 算数运算符

- + – * / ：加减乘除，与通常的用法相同
- MOD：取余，两数相除之余数
  - 例：10　　MOD 3 = 1
  - 例：10H　MOD 3 = 1H
- SHL：左移，按二进制左移
  - 例：10　　SHL 2 = 40
  - 例：14H　SHL 2 = 48H
  - 例：16H　SHL 2 = 58H
- SHR：右移，按二进制右移
  - 例：10　　SHR 2 = 2
  - 例：10H　SHR 2 = 4H
  - 例：16H　SHR 2 = 5H

三. 运算符号

2. 逻辑运算符

- AND：按位与
- OR：按位或
- XOR：异或
- NOT：非

76H AND 23H=22H
76H AND 23H=77H
76H XOR 23H=55H
NOT 76H =89H

三. 运算符号

3. 关系运算符

- **EQ**：等于（Equal）
- **NE**：不等于
- **LT**：小于（Less than）
- **LE**：小于或等于
- **GT**：大于（Greater than）
- **GE**：大于或等于

两个操作数必须是数值，或者是处在同一段内的存储器地址

4. 分析算符

- SEG ：取标号所处段的段基址
- OFFSET ：取标号存储位置与其所处段起始地址的距离，即标号的偏移地址
- TYPE ：取标号的类型值
- SIZE ：取DUP分配的内存单元的字节数
- LENGTH ：取DUP分配的内存单元数

DUP （duplicate）是一个操作符，在汇编语言中同db、dw、dd等一样，也是由汇编器识别处理的符号。

4. 分析算符

- SEG ：取标号所处段的段基址
- OFFSET ：取标号存储位置与其所处段起始地址的距离，即标号的偏移地址
- TYPE ：取标号的类型值

```
ALPHA DB    15,23,46
BETA  DW    4267H,25H
      MOV AX,SEG ALPHA
      MOV BX,SEG BETA
      MOV  CX,OFFSET ALPHA
      MOV  DX,OFFSET BETA
      MOV  SI,TYPE ALPHA
      MOV  DI,TYPE BETA
```

ALPHA	0FH	2000:0001H
	17H	2000:0002H
	2EH	2000:0003H
BETA	67H	2000:0004H
	42H	2000:0005H
	25H	2000:0006H
	00H	2000:0007H

DS = 2000H

4. 分析算符

- SIZE ：取DUP分配的内存单元的字节数
- LENGTH ：取与标号类型相关联的DUP分配的数据个数

```
ALPHA  DB    5 DUP(0)
BETA   DW    3 DUP (0)
GAMA   DD    4 DUP (0)
       MOV AX,SIZE ALPHA          ;5
       MOV BX,SIZE BETA           ;6
       MOV CX,SIZE GAMA           ;16
       MOV AX,LENGTH ALPHA        ;5
       MOV BX,LENGTH BETA         ;3
       MOV CX,LENGTH GAMA         ;4
```

5. 组合算符

- PTR　　：用来建立或临时改变存储器操作数的类型
  - 格式：NEWT PTR EXP
  - 说明：NEWT是要建或改变的标号或存储单元的新的类型，可以是BYTE，WORD等等。EXP是表达式，可以是已定义或未定义过的标号或间接寻址、基址寻址、变址寻址、基变址寻址方式表示的存储器操作数的存储单元。作用：防止二义性。

行号	程序指令		解释
1	MOV	WORD PTR[BX], 10H	；10H以字存放
2	MOV	BYTE PRT[BX], 10H	；10H以字节存放
3	ALPHA	DW 1234H	
4	MOV	AL, BYTE PTR ALPHA	；临时改变类型

5. 组合算符

- THIS　　：用来为一个标号建立一个新的类型
  - 格式：THIS TYPE_N
  - 说明：TYPE_N为类型符号名，可以是WORD，BYTE等等。用来建立一个新的标号的属性，该标号的段和偏移量就是下一个能分配的存储单元的段和偏移量。
  - 例子：

行号	程序指令		解释
1	**ADR1**	**EQU THIS WORD**	；ADR1为字类型
2	**ADR2**	**DB 15H, 26H, 37H**	；ADR1和ADR2有相同的段和偏移量
3	**MOV**	**AX, ADR1**	；AX = 2615H

6. 字节分离算符

- 用于将一个表达式的高字节和低字节分开。LOW取低字节，HIGH取高字节

```
MOV    AL,HIGH 1234H
MOV    AH,LOW 1234H
MOV    AL,LOW 12345
MOV    AL,LOW 34*97+8
MOV    AL, LOW OFFSET A1
```

三. 运算符号

7. 记录专用算符（暂时不讲）

- MASK
- WIDTH

8. 算符优先级

- 表达式求值时，各种运算符的优先次序(由高至低)如下：
  1) 括号内的表达式
  2)LENGTH，SIZE，WIDTH，MASK·
  3)PTR，OFFSET，SEG，TYPE·
  4)HIGH，LOW·
  5) *， / ，MOD，SHL，SHR·
  6)+，–·
  7)EQ，NE，LT，LE，GT，GE·
  8)NOT·
  9)AND·
  10)OR，XOR。
- 优先级别相同时，按出现次序由左至右顺序计算

第四章 汇编语言

四. 伪指令

1. 符号定义伪指令
2. 数据定义伪指令
3. 段定义伪指令
4. 其他伪指令

伪指令：伪指令是指汇编程序（汇编器）对源程序汇编时起说明作用的命令

四. 伪指令

1. 符号定义伪指令

- EQU：等价伪指令
- =　　：等号伪指令

汇编语言中有两种符号：

- 汇编程序已经规定且有一定意义的符号，称为保留符号，如机器指令的助记符号、伪指令的助记符号、寄存器名字、各种运算符等
- 编程人员（用户）自定义的符号

四. 伪指令

1. 符号定义伪指令

- **EQU**：等价伪指令
  - 格式：SN EQU EXP
  - 说明：EQU伪指令用来定义一个符号，用该符号代表机器指令助记符、伪指令助记符、寄存器名字、各种运算符号、常数等。
  - 示例：

行号	程序指令		解释
1	VAL	EQU 100	；VAL值为100
2	NUM	EQU VAL+50	；NUM值为150
3	OPA	EQU ADD	；替代ADD指令
4	ADR	EQU [SI+BX+1000H]	；替代寻址方式

四. 伪指令

1. 符号定义伪指令

- =：等价伪指令
  - 格式：SN = EXP
  - 说明：EXP可以是表达式、各种常数和符号数据。
  - 示例：

行号	程序指令		解释
1	**VAL**	**= 100**	；VAL值为100
2	**VAL**	**= 8*5**	；VAL值为40
3	**VAL**	**= VAL+5**	；VAL值为45

注意：
- =伪指令只能代表表达式或常数，不能是指令助记符等
- EQU定义的符号不能重定义，而=定义的可以

2. 数据定义伪指令

- 用来定义一个存储单元的符号名，并初始化该单元或由该单元开始的若干连续单元
- 初始化单元就是将一个确定的数值或不确定的值（在定义语句中用？表示）放入指定的内存单元
- 数据定义伪指令有五种：
  - DB：字节定义伪指令
  - DW：字定义伪指令
  - DD：双字定义伪指令
  - DQ：四字定义伪指令
  - DT：十字定义伪指令

2. 数据定义伪指令

- DB：字节定义伪指令
  - 格式：[SN] DB EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于8位二进制数能表示的最大值
    - 数据书写形式中除科学表示法外的其他形式
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 例子：

FIRST	03H	000DH
	12H	000EH
	25H	000FH
	41H	0010H
	42H	0011H
	43H	0012H
SECOND	2DH	0013H
	0CH	0014H
	1FH	0015H
	06H	0016H
THIRD	??	0017H
	??	0018H
	36H	0019H
	FFH	001AH
	1EH	001BH

```
000D  03 12 25 41 42 43          FIRST   DB      3,18,25H,'ABC'
0013  2D 0C 1F 06                SECOND  DB      3*15,48/4,36-5,26 MOD 10
0017  ?? ?? 36 FF 1E             THIRD   DB      ?,?,36H,-1,36Q
001C  0005[1A]                   FOUTH   DB      5 DUP (26)
0021  0002[0002[04] 0F] 7F]      FIVET   DB      2 DUP (2 DUP (4),15),7FH
0027  0003[??]                   SIXTH   DB      3 DUP (?)
```

2. 数据定义伪指令

- DB：字节定义伪指令
  - 格式：[SN] DB EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于8位二进制数能表示的最大值
    - 数据书写形式中除科学表示法外的其他形式
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 例子：

```
000D  03 12 25 41 42 43        FIRST   DB      3,18,25H,'ABC'
0013  2D 0C 1F 06             SECOND  DB      3*15,48/4,36-5,26 MOD 10
0017  ?? ?? 36 FF 1E          THIRD   DB      ?,?,36H,-1,36Q
001C  0005[1A]                FOUTH   DB      5 DUP (26)
0021  0002[0002[04] 0F] 7F]   FIVET   DB      2 DUP (2 DUP (4),15),7FH
0027  0003[??]                SIXTH   DB      3 DUP (?)
```

FOUTH	1AH	001CH
	1AH	001DH
	1AH	001EH
	1AH	001FH
	1AH	0020H
FIVET	04H	0021H
	04H	0022H
	0FH	0023H
	04H	0024H
	04H	0025H
	0FH	0026H
	7FH	0027H
SIXTH	??	0028H
	??	0029H
	??	002AH

2. 数据定义伪指令

- **DW**：字定义伪指令
  - 格式：[SN] DW EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于16位二进制数能表示的最大值
    - 数据书写形式中除科学表示法外的其他形式
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 每个数据项占用两个字节存储单元，数据高位在高地址
  - 例子：

```
002E   1234 8000 5859    FSTW DW   1234H,-32768,'XY'
0034   1000 0046 FFE7    SCDW DW   256*16,46H,-25
003A   ???? 002E R 00FF  TRDW DW   ?,OFFSET FSTW,255
0040   0003[0510]        FUTW DW   3 DUP (510H)
```

FSTW	34H	002EH
	12H	002FH
	00H	0030H
	80H	0031H
	59H	0032H
	58H	0033H
SCDW	00H	0034H
	10H	0035H
	46H	0036H
	00H	0037H
	E7H	0038H
	FF	0039H
TRDW	??	003AH
	??	003BH
	2EH	003CH
	00H	003DH

2. 数据定义伪指令

- **DW**：字定义伪指令
  - 格式：[SN] DW EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于16位二进制数能表示的最大值
    - 数据书写形式中除科学表示法外的其他形式
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 每个数据项占用两个字节存储单元，数据高位在高地址
  - 例子：

FUTW	FFH	003EH
	00H	003FH
	10H	0040H
	05H	0041H
	10H	0042H
	05H	0043H
	10H	0044H
	05H	0045H

```
002E   1234 8000 5859     FSTW DW   1234H,-32768,'XY'
0034   1000 0046 FFE7     SCDW DW   256*16,46H,-25
003A   ???? 002E R 00FF   TRDW DW   ?,OFFSET FSTW,255
0040   0003[0510]         FUTW DW   3 DUP (510H)
```

四. 伪指令

2. 数据定义伪指令

- DD：双字定义伪指令
  - 格式：[SN] DD EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于32位二进制数能表示的最大值
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 每个数据项占用四个字节存储单元，数据高位在高地址，EXP可以是十进制科学表示法的数据
  - 例子：

DFST	FFH	0052H
	FFH	0053H
	FFH	0054H
	FFH	0055H
	00H	0056H
	00H	0057H
	00H	0058H
	80H	0059H
DSCD	00H	005AH
	80H	005BH
	12H	005CH
	42H	005DH
	00H	005EH
	80H	005FH
	12H	0060H
	C2H	0061H

```
0052  FFFFFFFF00000080    DFST   DD    4294967295,80000000H
005A  00801242008012C2    DSCD   DD    36.625,-0.36625E+2
0062  0002[FF030000]      DTRD   DD    2 DUP (1023)
006A  4E4D0000            DFUT   DD     'MN'
006E  0052 ---- R         DFIV   DD     DFST
```

四. 伪指令

2. 数据定义伪指令

- DD：双字定义伪指令
  - 格式：[SN] DD EXP
  - 说明：SN为可选项，EXP为表达式，其值小于或等于32位二进制数能表示的最大值
    - N DUP(EXP)，n为重复次数，EXP为表达式
    - 问号（？），表示用不确定的值初始化内存单元
  - 每个数据项占用四个字节存储单元，数据高位在高地址，EXP可以是十进制科学表示法的数据
  - 例子：

DTRD	FFH	0062H
	03H	0063H
	00H	0064H
	00H	0065H
	FFH	0066H
	03H	0067H
	00H	0068H
	00H	0069H
DFUT	4EH	006AH
	4DH	006BH
	00H	006CH
	00H	006DH
DFIV	52H	006EH
	00H	006FH
	××	0070H
	××	0071H

```
0052  FFFFFFFF00000080    DFST   DD    4294967295,80000000H
005A  00801242008012C2    DSCD   DD    36.625,-0.36625E+2
0062  0002[FF030000]       DTRD  DD     2 DUP (1023)
006A  4E4D0000             DFUT   DD      'MN'
006E  0052 ---- R          DFIV   DD      DFST
```

3. 段定义伪指令

- 编程时的代码段、数据段、附加段和堆栈段都是逻辑段，由段定义伪指令来定义，经汇编程序（汇编器）连接后转换为真正的内存物理地址
- 段定义伪指令有四种：
  - SEGMENT：段首定义伪指令
    - 格式：SNAME SEGMENT [PRMT1][PRMT2][‘PRMT3’]
    - 说明：告诉汇编程序SNAME段由此开始
  - ENDS　　：段结束伪指令
  - ASSUME　：段假定伪指令
    - 格式：ASSUME SR：SNAME [, SR：SNAME]
    - 说明：段说明，让汇编程序知道段寄存器应该有段基址，而段寄存器实际值需要在程序中显示说明

3. 段定义伪指令

```
CSEG        SEGMENT
            ASSUME    CS:CSEG,DS:DSEG
            MOV       AX,DSEG
            MOV       AL,DATA1
            ADD       AL,DATA2
            MOV       SUM,AL
            HLT
CSEG        ENDS
```

4. 代码定位伪指令 ORG

格式：ORG　EXP

说明：EXP 是数学表达式，其值为一个无符号的 16 位二进制数。此语句的作用是确定其后的数据和代码存放在相应段中的起始位置。EXP 的值是相对于段基址的偏移量。以后的数据或代码顺序连续存放，除非遇到新的 ORG 伪指令。

例：

```
MY_DATA     SEGMENT                              ;数据段开始
ALPHA       DB          15, 23, -5
BUFF        DB          10 DUP(0)
MY_DATA     ENDS                                 ;数据段结束
MY_CODE     SEGMENT                              ;代码段开始
            ASSUME      CS: MY_CODE, DS: MY_DATA
            ASSUME      ES: MY_DATA
START:      MOV         AX, MY_DATA              ;对 DS, ES 赋值
            MOV         DS, AX
            MOV         ES, AX
            ...
MY_CODE     ENDS                                 ;代码段结束
            END         START                    ;源程序结束
```

4. 代码定位伪指令 ORG

```
COMN_SEG   SEGMENT                              ;段开始
           ORG      100H
           JMP      BEGIN
ALPHA      DW       1235H, 36H, 45H
BETA       DB       ' HOW ARE YOU?'
BUFFER     DW       10 DUP( 0)
           ASSUME   CS: COMN_SEG, DS: COMN_SEG  ;段说明
           ASSUME   ES: COMN_SEG, SS: COMN_SEG
BEGIN:     MOV      AX, COMN_SEG
           MOV      DS, AX
           MOV      ES, AX
           MOV      SS, AX
           . . .
COMN_SEG   ENDS                                 ;段结束
           END      BEGIN
```

5. 其他定义伪指令

- EVEN　　是程序计数器定位在偶地址单元。
- .RADIX　　改变默认进制。
- NAME　　为程序模块命名。
- END　　源程序结束。

第四章 汇编语言

五. 汇编语言的上机过程

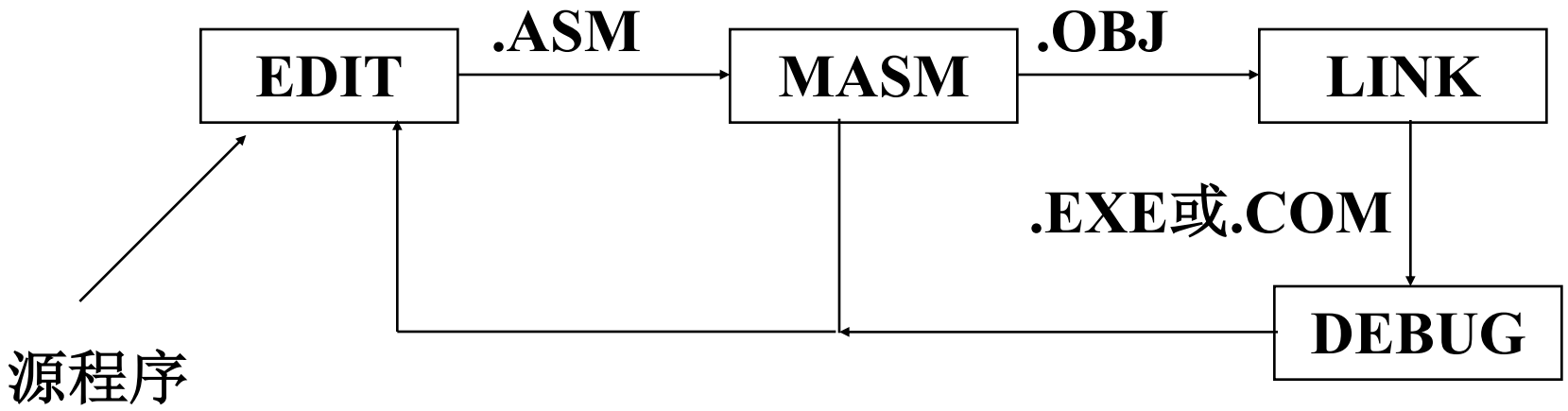
1. 汇编语言的工作环境
2. 上机过程

五. 汇编语言的上机过程

1. 汇编语言的工作环境

- 编辑程序：EDIT　NOTEBOOK
- 汇编程序：MASM.EXE　TASM.EXE
- 连接程序：LINK.EXE TLINK.EXE
- 调试程序：DEBUG.COM(EXE)

2. 上机过程



随堂测验

写出下列表达式的值：

(1) 70H SHR 4/7

(2) 18H SHL 2 MOD 5 + 9

(3) NOT (79H AND 3AH OR 38H)

(4) (75H XOR 49H) AND (102 GT 51)