用汇编语言编写程序

许 向 阳

xuxy@hust. edu. cn

华中科技大学 计算机科学与技术学院

MASM、Visual Studio 2019 等环境下，汇编语言源程序的开发

不同的编译器支持的语法不同！

汇编语言源程序举例

求：1+2+3+…+100之和，结果放入eax中。

和：　eax

加数：ebx

```
eax=0;
for (ebx=1;ebx<=100;ebx++)
   eax = eax+ebx;
```

```
      eax=0;
      ebx=1;
lp: if (ebx>100)
          goto etit;
      eax=eax+ebx;
      ebx=ebx+1;
      goto  lp;
exit:
```

工程：asm_01_01

汇编语言源程序举例

例1 ：求 1+2+3+...+100 之和，放在eax中，并显示。

```
.686P
.model flat, c
   ExitProcess proto stdcall :dword
   includelib  kernel32.lib
   printf      proto c :vararg
   includelib  libcmt.lib
   includelib  legacy_stdio_definitions.lib
.data
 lpFmt    db "%d",0ah, 0dh, 0
.stack    200
.code
```

汇编语言源程序举例


```
main proc
; eax=0; for (ebx=1;ebx<=100;ebx++) eax=eax+ebx;
    mov   eax, 0       ;  (eax)=0，用于存放累加和
    mov   ebx, 1       ;  (ebx)=1，用于指示当前的加数
lp: cmp   ebx, 100     ; 连续两条指令，等同于
                       ;  if (ebx>100) goto exit
     jg    exit
     add   eax, ebx    ;  (eax) = (eax) + (ebx)
     inc   ebx         ;  (ebx) = (ebx) +1
     jmp   lp          ;  goto lp
exit:
    invoke printf, offset lpFmt, eax
    invoke ExitProcess, 0
main  endp
end
```

汇编语言源程序举例

例2：输入5个有符号数到一个数组缓冲区中，然后求它们的最大值并显示该最大值。

工程：**asm_01_02**

汇编语言源程序举例


```
.686P
.model flat, c
  ExitProcess proto stdcall :dword
  includelib  kernel32.lib
  printf      proto c :ptr sbyte, :vararg
  scanf       proto c :ptr sbyte, :vararg
  includelib  libcmt.lib
  includelib  legacy_stdio_definitions.lib
.data
  lpFmt    db   “%d”,0ah, 0dh, 0   ;输出格式串
  buf      db   “%d”, 0    ; 输入格式串中无 \n
  x        dd   5 dup(0)    ;  int x[5];
```

汇编语言源程序举例


```
.stack 200
.code
main  proc
    ; 输入5个数
    ;    ebx=0;
    ;    do { scanf("%d",&x[ebx*4]); ebx++; }
    ;    while (ebx!=5)
   mov   ebx, 0
input_5num:
   invoke scanf, offset buf, addr [x+ebx*4]
   inc    ebx
   cmp    ebx, 5
   jne    input_5num
```

汇编语言源程序举例


```
; 求最大数,先将第一个数作为最大数，放在eax中。
    mov   eax, x[0]          ; eax = x[0]
    mov   ebx, 1             ; ebx = 1
lp: cmp   ebx, 4             ; if (ebx>4) goto exit
    jg    exit               ; 结束找整个最大数的循环
    cmp   eax, x[ebx*4]      ; (eax)比当前数大或者相等,
                             ;   则不要做任何处理
    jge   next
    mov   eax, x[ebx*4]      ; eax = x[ebx*4]
next: add  ebx, 1            ; ebx = ebx+1
    jmp   lp                 ; goto lp
exit:
    invoke printf, offset lpFmt,eax
    invoke ExitProcess, 0
main  endp
end
```

数据段定义

数据段定义

1. 数据定义伪指令

[变量名]　数据定义伪指令　表达式[，…]

db、byte、sbyte	字节类型
dw、word、sword	字类型
dd、dword、sdword	双字类型
dq、qword、sqword	四字类型
dt、tbyte	10个字节类型
real4	单精度浮点数类型
real8	双精度浮点数类型

表达式有4种形式

数据段定义

2. 表达式

[变量名]　数据定义伪指令　表达式[，…]

数据定义伪指令：DB，DW，DD,DQ，SBYTE ……

表达式的4种形式

- 数值表达式
- 字符串
- 地址表达式
- 重复子句 n DUP（表达式[，…]）

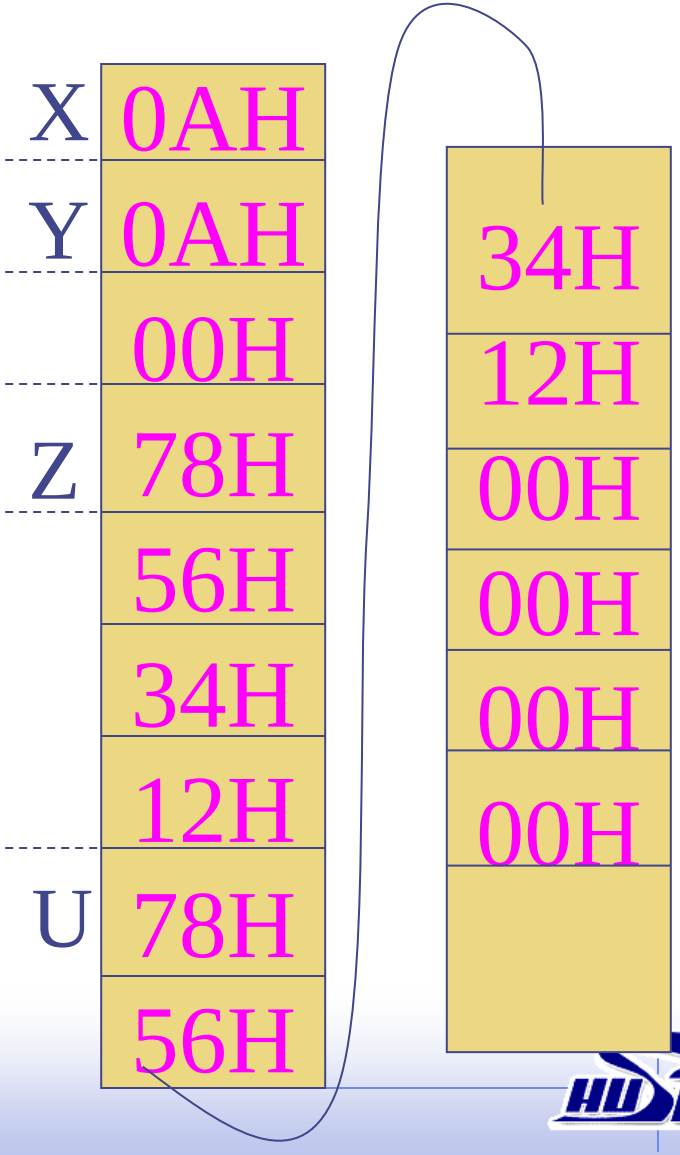
数据段定义

- 数据定义伪指令　数值表达式

```
X   DB    10
Y   DW    10
Z    DD    12345678H
U    DQ    12345678H
V    DT    12345678H
```

如下定义正确吗？

~~K DB 1234H~~



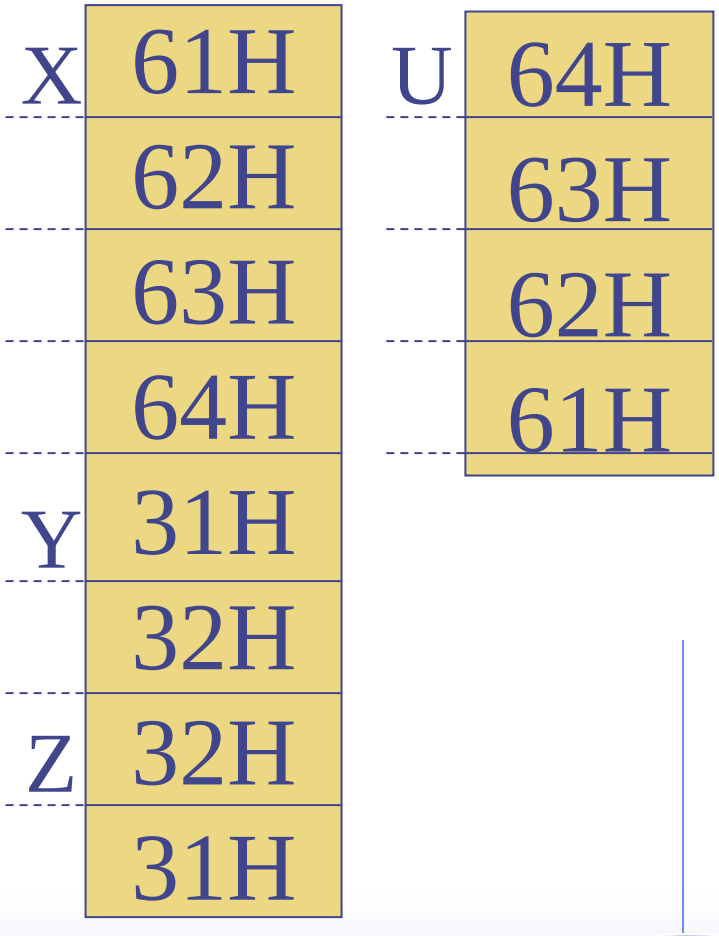
数据段定义

- 数据定义伪指令　字符串

```
X  DB  ‘abcd’
Y  DB  ’12’
Z  DW  ’12’
U  DD  ‘abcd’
```

如下定义正确吗？

~~K DW ‘1234’~~



数据段定义

- 数据定义伪指令　地址表达式

地址表达式：由变量、标号、常量、[R] 和运算符组成的有意义的式子。

在数据定义语句中，不能出现带寄存器符号的地址表达式。

数据段定义

■数据定义伪指令　地址表达式

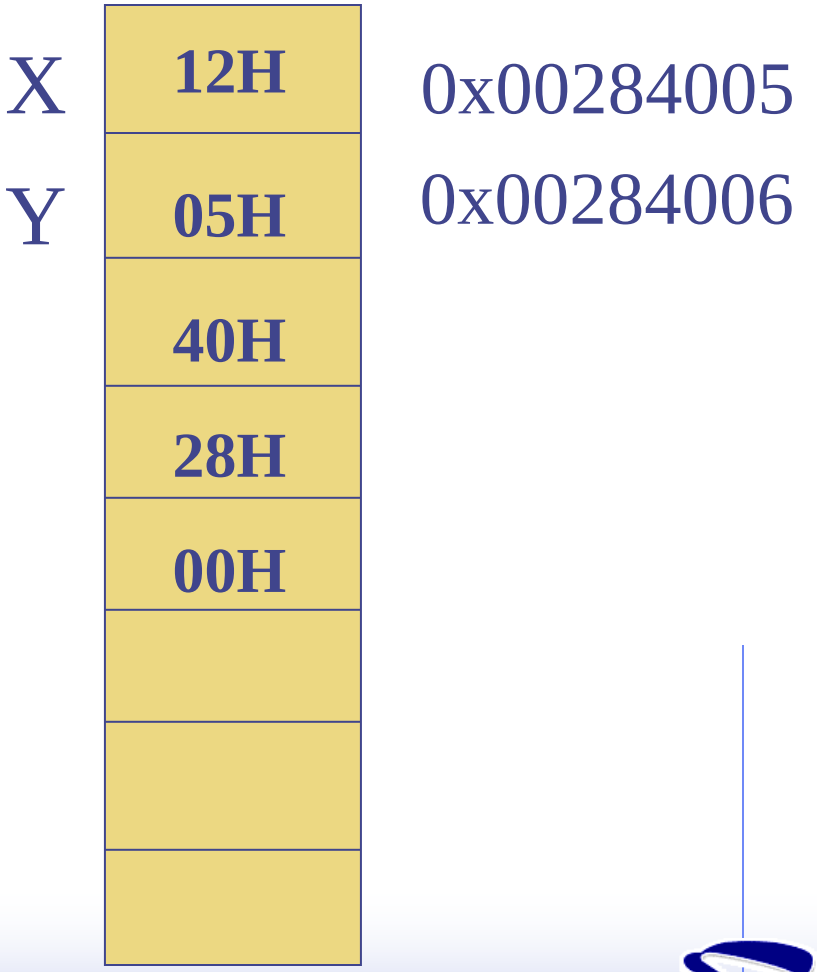
➢ **DD 变量或标号**

变量定义在32位段中，偏移地址是32位的，用变量的偏移地址来初始化相应双字单元。

数据段定义

地址表达式的数据存放

```
.DATA
X   DB    12H
Y    DD    X
```



数据段定义

- 数据定义伪指令　重复子句

  N DUP (表达式[，表达式]…)

  例1：　X DB 3 DUP (2)

  　　X DB 2, 2, 2

数据段定义

■数据定义伪指令　重复子句

例2：　Y　DB　3 DUP (1,2)

Y　DB　1, 2, 1, 2, 1, 2

例3：Z　DB　3 DUP (1, 2 DUP (2))

Z　DB　3 DUP (1, 2 ,2　)

Z　DB　1, 2, 2, 1, 2, 2, 1, 2, 2

数据段定义

Q：同一个数值表达式在不同数据定义伪指令后的表现形式？

Q：一条数据定义伪指令后的多个表达式的存放方法？

Q：地址表达式是什么？

Q：地址表达式出现在 DD后, 其意义是什么？

Q：重复子句的嵌套和展开？

数据段定义


```
buf     dd   10, 20, 30, 40, 50
  x     db   60H
  y     dw   60H
  z     dd   BUF
```

地址: 0x00384005

```
0x00384005  0a 00 00 00 14 00 00 00 1e 00 00 00 28 00 00 00 32 00 00 00 60 60 00 05
0x0038401D  40 38 00 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00384035  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0038404D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00384065  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

监视 1

名称	值	类型
&buf,x	0x00384005 {LOCAL_V.exe!unsigned long buf} {0x0000000a}	unsig
&x,x	0x00384019 "``"	unsig
x	96 '`'	unsig
y,x	0x0060	unsig
&y,x	0x0038401a {LOCAL_V.exe!unsigned short y} {0x0060}	unsig

断点

新建

名称

- c2_address2.asm，行 2
- c_example.cpp，行 15
- LOCAL_VAR.ASM，行 2
- LOCAL_VAR.ASM，行 3
- local_variable.asm，行

3. 汇编地址计数器

用来记录当前拟分配的存储单元的地址

```
x  db  'ABCD'
y  dw  $-x              ;  4
z  dw  $-x,  $-x        ;  6,  8

buf   dw  20,  30,   40
len = ($-buf)/2    ;该值为buf中字数据的个数。
```

数据段定义

- 使用置汇编地址计数器伪指令来修改$的值

  org 数值表达式

  功能：将$置为数值表达式的值。

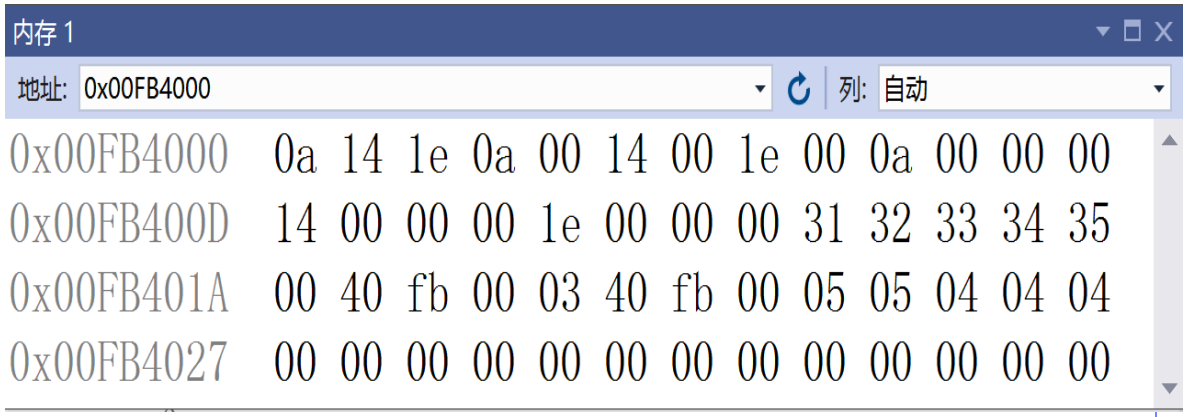
  例：　org $+5 ；

  　　在上一个变量空间分配后，空出5个字节。

- 使用伪指令 even、align，来指定下一个变量的起始地址的对齐特性，其本质是修改$的值。
- even是使$变成大于等于原$的最小偶数。
- align 后面跟1、2、4、8、16等数，使得$变成不小于原$且能被1、2、4、8、16整除的数。

数据段定义

```
.686P
.model flat, stdcall
  ExitProcess proto  :dword
  includelib  kernel32.lib
.data
  x  db  10, 20, 30
  y  dw  10, 20, 30
  z  dd  10, 20, 30
  u  db  '12345'
  u_len  = 5
  p  dd  x,  y
  q  db  2 dup (5), 3 dup (4)
.stack 200
.code
 start:
    invoke ExitProcess, 0
end start
```



数据段定义

x	0aH	00FB4000H
	14H	
	1eH	
y	0aH	00FB4003H
	00H	
	14H	
	00H	
	1EH	
	00H	
z	0aH	00FB4009H
	00H	
	00H	
	00H	
	14H	00FB400DH
	00H	
	00H	
	00H	

	0eH	00FB4011H
	00H	
	00H	
	00H	
u	31H	00FB4015H
	32H	
	33H	
	34H	
	35H	
p	00H	00FB401AH
	40H	
	0fbH	
	00H	
	03H	00FB401EH
	40H	
	0fbH	
	00H	

指令语法规则

指令语法注意事项

一般情况下，指令的共同要求：

(1) 双操作数的操作数类型必须匹配。

MOV AX , BH

SUB BL, CX

MOV SI , CL

ADD BUF , AX ； 其中 BUF DB 1, 2

MOV ds:[2000] ,2

指令语法注意事项

一般情况下，指令的共同要求：

(1) 双操作数的操作数类型必须匹配。

(2) 目的操作数一定不能是立即操作数

CMP AX，2

功能是：根据 (AX) –2 的结果设置标志位

CMP 2，AX

指令语法注意事项

一般情况下，指令的共同要求：

(1) 双操作数的操作数类型必须匹配。

(2) 目的操作数一定不能是立即操作数。

(3) **目的操作数和源操作数不能同时为存储器操作数。** 如果一个操作数在数据存储单元中，另一个一定要是立即数和寄存器操作数。

~~MOV BUF1, BUF2~~
~~ADD WORD PTR [ESI], [EDI]~~
~~SUB BUF1, [ESI]~~

指令语法注意事项

程序中的伪指令

- 处理器选择伪指令
- 存储模型说明伪指令
- 数据定义伪指令
- 符号定义伪指令
- 段定义伪指令
- 过程定义伪指令
- 程序模块的定义与通讯伪指令
- 宏定义伪指令
- 条件汇编伪指令
- 格式控制、列表控制及其它功能伪指令

处理器选择伪指令

.686　　接受Pentium Pro指令，特权指令除外
.686P　接受全部Pentium Pro指令
.MMX　　接受MMX指令
.XMM　　接受SSE、SSE2、SSE3指令

一般处理器选择伪指令采用：

.686P
.XMM

存储模型说明伪指令

- .model 存储模型 [,语言类型]
- 对于Win32程序，存储模型选择为 flat
  代码和数据全部放在同一个4G空间内；
- “语言类型”指定了函数参数的传递方法和释放参数所占空间的方法；
- 语言类型为 stdcall 或者 c
- 函数原型描述中最右边的参数最先入栈、最左边的参数最后入栈；
- stdcall 被调用者（即函数内）在返回时释放参数占用的堆栈空间；
- c 在调用函数中释放参数所占的空间。

段定义及程序结束

简化的段定义

- 一个段的开始也是前一个段的结束。
- 定义数据段伪指令：.data　或 .data?
- 定义代码段伪指令：.code　[段名]
- 定义堆栈段伪指令：.stack [堆栈字节数]
  省略时为1024
- 常数（只读）数据段定义：.const
- 程序结束伪指令：end　[表达式]

段定义及程序结束

完整的段定义

段名 SEGMENT [READONY][使用类型]
[定位方式] [组合方式]
[特性] [类别名]

.....

段名 ENDS

使用类型：USE16、USE32、FLAT。

对于 .386及以上的处理器，默认为 USE32

段定义及程序结束

定位方式：段从何处开始。

BYTE, WORD, DWORD, PARA, PAGE

起始地址被 1、2、4、16、256 整除。

缺省值为 PARA

段定义及程序结束

组合方式：本段与其他段如何组合。

不选择、PUBLIC， STACK， COMMON， AT 表达式，MEMORY、PRIVATE

不选择：独立的段，不组合，缺省的方式；

PUBLIC：同名、同类别的段顺序连接，组合成一个段；

STACK： 对堆栈段组合成一个堆栈段；

COMMON：覆盖，与同名同类别的段具有相同的段首址；

AT 表达式：本段装配在指定位置；

MEMORY ：本段定位在所有连接在一起的段之上（高地址）

段定义及程序结束

特性：本段的属性。

INFO、 READ、 WRITE、 EXECUTE、 SHARED、 NOPAGE、 NOCACHE、DISCARDED

类别：‘DATA’、‘CODE’、‘CONST’、‘STACK’。

物以类聚、人以群分。

相同类别的段在空间上相邻，但又各自独立。

段定义及程序结束


```
.686P
ExitProcess proto stdcall:dword
includelib  kernel32.lib
printf      proto c :ptr sbyte, :vararg
includelib  libcmt.lib
includelib  legacy_stdio_definitions.lib
seg1 segment use32
     lpFmt db "%d", 0ah, 0dh
seg1 ends

seg2 segment stack use32
      db 200 dup(0)
seg2 ends
```

段定义及程序结束


```
seg3 segment use32 execute
     assume cs:seg3, ss:seg2, ds:seg1
main proc c
     mov eax, 0
     mov ebx, 1
  lp:cmp ebx,100
     jg  exit
     add eax, ebx
     inc ebx
     jmp lp
 exit: invoke printf, offset lpFmt, eax
       invoke ExitProcess, 0
main endp
seg3 ends
end
```

段定义及程序结束

16位段程序示例

运行环境：DOSBOX、MASM60

在 DOSBOX 下，用 masm汇编，用link链接，用TD调试

段定义及程序结束


```
.386
DATA SEGMENT USE16
lpFmt db    0ah, 0dh, "$"
  X   DB  10, 255, -1
  Y   DW  10, 255, -1
  Z   DD  10, 255, -1
  U   DW  ($-Z)/4
  STR1 DB 'Good', 0
  P   DD  X,   Y
  Q   DB   2 DUP (5, 6)
  buf1 DB  '00123456789','$'   ;结束符号为$
  buf2 DB  12 dup(0)
DATA ENDS
```

段定义及程序结束


```
STACK   SEGMENT   USE16   STACK
    DB   200 DUP(0)
STACK   ENDS
CODE   SEGMENT   USE16
    ASSUME   CS:CODE,SS:STACK,DS:DATA
START: MOV   AX,DATA
       MOV   DS,AX
    MOV   ESI,OFFSET buf1
    MOV   EDI,OFFSET buf2
    MOV   ECX,0
L1:
    MOV   EAX, [ESI]
    MOV   [EDI],EAX
```

段定义及程序结束

華中科技大學

```
    ADD  ESI, 4
    ADD  EDI, 4
    ADD  ECX, 4
    CMP  ECX,12
    JNZ  L1
    MOV  DX, OFFSET buf1    ;DOS功能调用显示字符串
    MOV  AH,9
    INT   21H
    MOV  DX, OFFSET  lpFmt
    MOV  AH,9
    INT   21H
    MOV  DX, OFFSET buf2
    MOV  AH,9
    INT   21H
    MOV AH,4CH
    INT  21H
CODE ENDS
  END START
```

程序设计

设计一个程序要点：

- 认真分析问题的需求，选择好解决方法；
- 针对选定的算法，编写高质量的程序。

高质量的程序：

- 满足设计的要求。
- 结构清晰、简明、易读、易调试。
- 执行速度快。
- 占用存储空间少。

程序设计

汇编语言程序设计的一般步骤：

（1）分析问题，选择合适的解题方法

（2）分配寄存器和存储区(变量命名)

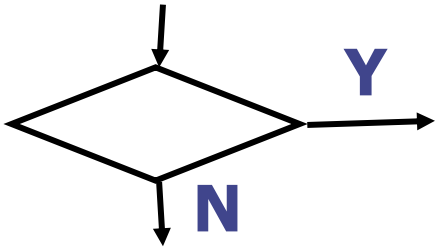
（3）绘制程序的流程图

（4）根据流程图编写程序

（5）静态检查与动态调试

几种框图符号

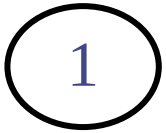
起始、终止框



判断框

处理说明框

子程序调用框



连接框

流向线

VS2019 下的汇编程序开发

许向阳.《x86汇编语言程序设计》. 华中科技大学出版社
第 19章 上机操作

见 QQ 群里的课件：x86汇编语言程序设计_图书PDF.rar

VS2019 下的汇编程序开发

① 新建一个空项目

② 配置新项目

③ 设置生成依赖项

④ 添加汇编源程序文件

⑤ 编辑源程序

⑥ 生成执行程序

按步骤，顺序做！