目录

先总结一下汇编个人觉得的考点：

考法无非两种：读程序、写完整的程序

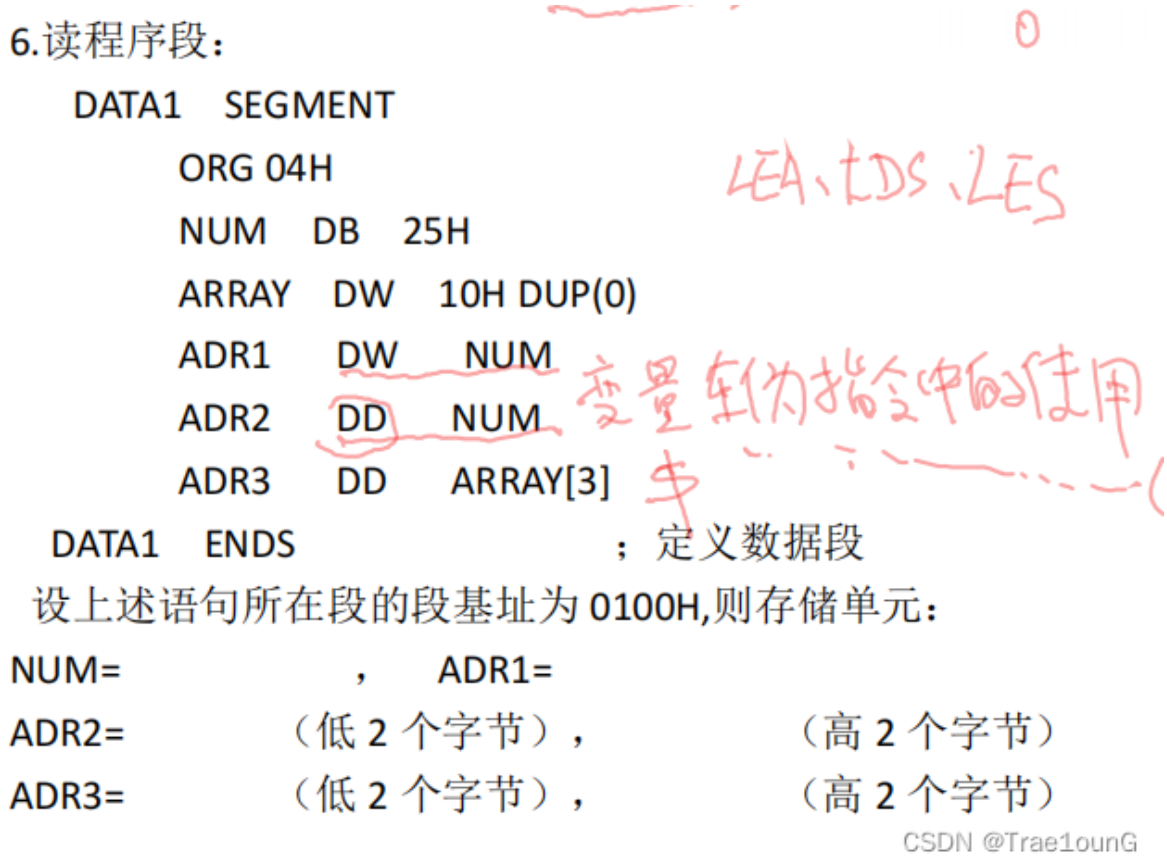
读程序个人猜想主要的考法

代码段中变量的内容

涉及到段内每个字节的存储

重点需要掌握：**org设置偏移地址；dw、dd在段内的存储方式；变量在伪指令中的含义**【dw偏移地址，dd低位偏移高位段地址】（当然在指令里面的也很重要）；**[]的作用**【如Array[3] 即在Array的偏移地址上加3再访问】；$的含义

例题：



程序执行中寄存器的内容

这个就和写完整的程序差不多了，需要对各个指令的掌握比较扎实，当然简单很多毕竟不用真正的掌握完整程序的编写。

贴几道例题，画画图分析一下

例一：

已知AX=0078H，BX=0408H

```
SHR  BX，1
 AND  BX，0F0FH
 MOV  CX，4
 LOP：XCHG   BH，BL
    DIV  BL
    SAL  AX，1
    LOOP  LOP

上述程序段执行后，AX= BX=
① 30和0402H ② 30和 0204H
③ 35和0400H ④ 35和 0400H
```

> 答案：2，
> 注意SAL示意图
> XCHG交换指令

例二

VARY1 EQU BYTE PTR VARY2
VARY2 DW 0ABCDH
……
SHL VARY1,1
SHR VARY2,1

上述两条指令执行后,VARY2字存储单元内容是

思考区

---

① 0ABCDH ② 0BCDEH ③ 55CDH ④ 0AB55H

> 答案：3
> 注意VARY1等价于VAR2的字节单元，也就是VARY1=CDH，在“SHL VARY1,1”进行的是字节为单元的移位（只对CDH进行操作），“SHR VARY2,1”是对上一步移位后的结果再次进行SHR操作。注意示意图需画对！

例三：

已知AX=0FFFFH，DX=0001H

MOV CX，2
LOP：SHL AX，1
RCL DX，1
LOOP LOP

上述程序段执行后DX= AX=
① DX=0006H AX=0FFFDH ② DX=0006H AX=0FFFCH
③ DX=0007H AX=0FFFDH ④ DX=0007H AX=0FFFCH

思考区

> 答案：4

例四：

ARRAY DW 6,9,+ 4 , 10 H , +4,10H,+4,10H,+4
……
MOV AX,ARRAY+4
设变量ARRAY的偏移量是0084H，上述指令执行后AX中的内容是
① 0009H ② 008CH ③ 0090H ④ 0010H

> 答案：2
> $+4就是把当前地址的值加上4赋给该存储单元
> 该题要注意ARRAY是以字为单位的！不能看到6，,9想当然以为是字节

注：以上例题大多来自 https://blog.csdn.net/weixin_53937982/article/details/118311947#comments_23760209 可以参照上面的题算一算。这里仅总结作为参考。

写完整程序的方法
这一段最多只能针对完全没有学懂汇编，不知道怎么考试的人。这里写一个详细的模板，如果实在考试写不出来，基本框架至少能得分。

```
data segment
    ;这是注释,考试逻辑清楚的注释或许能加分
    ;定义题目上给的变量和你需要的变量
    ;如
    num1 db ?
data ends
;必须要的堆栈段
stack1 segment stack
   ;初始化堆栈段
    dd 20h dup(0);这一段随意
    ;dw 40h dup(0)都行，我也没看到固定写法
stack1 ends
code segment
    assume cs:code,ds:data,ss:stack1
    start:
;必须要有开头标记
;很重要的一点，作者差点忘记，双操作数中间要加逗号
;mov ax,bx要写，int 21h不写

mov ax,data
mov ds,ax ;初始化数据段，必须有

...
;其他需要的操作
;如果有要求重复n次的
;可以写 mov cx,n
;如果要求读入数字的
;可以写 mov ah,1
;int 21 h
;要求显示的
;mov dl xx
```

```
;mov ah 2
;int 21h
...
;其他处理逻辑，这些就得自己会才行了

;调用子程序
call subseg
;程序执行完
mov ah 4ch
int 21h;返回dos系统
;如果要求有子程序，可以写
subseg proc
    push ax
    push bx
    ;如果不知道要保护哪些，push就完了
    ...
    ;反着来就完了
    pop bx
    pop ax
    ret;必须有
subseg endp;必须有

code ends;写在最后
    end start;这里写开头的标记
```

在外面标注一下,需要写的代码部分

```
data segment

num1 db ?

data ends

stack1 segment stack

·   dd 20h dup(0)

stack1 ends
code segment
    assume cs:code,ds:data,ss:stack1
    start:
mov ax,data
mov ds,ax ;初始化数据段，必须有
code ends;写在最后
    end start;这里写开头的标记
```

这里可以给出几个练手的题目

1.试编写一完整源程序：数据段中 BUFFER 数据区存放有,16 个无符号字节数，编程将其中第 2、5、9、14、15 个字节内容加 5，其余字节内容乘 3（假定运算不会溢出）。设数据段中相应的定义如下：（8分）

BUFFER DB 12, 18, 34, 57, 7,21,60,41,40,30,10,20,3,10,27,42

注：该题要求编写完整的数据段、堆栈段及代码段的定义，以及程序结束并返回操作系统的语句。

（中等）

2.编写一程序段，把从 BUFFER 开始的 100 个字节的内存区域初始化成 55H、0AAH、55H、0AAH、•••••、55H、0AAH。


```
STACK SEGMENT
DW 64 DUP(?)
STACK ENDS
DATA SEGMENT
BUFF DB 100 DUP(?)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
XOR CX,CX
MOV CX,32H
MOV DI,OFFSET BUFF
A1:
MOV [DI],55H
MOV [DI+01H],0AAH
INC DI
INC DI
DEC CX
JNE A1
A2:XOR AX,AX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

（中等）分别统计下列20个数中大于5、小于零和大于零且小于等于5的数据个数，分别存入字节单元RES1、RES2和RES3中。

```
BUFF DB -1, 20, 3, 30, -5, 15, 100, -54, 0, 4, 78, 99, -12, 32, 3, 23, -7, 24, 60,-51

DATA SEGMENT
  BUF DB -1,20,3,30,-5,15,100,-54,0,4,78,99,-12,32,3,23,-7,24,60,-51
  RES1 DB ?
  RES2 DB ?
  RES3 DB ?
DATA ENDS
STACK1 SEGMENT PARA STACK
  DW 20H DUP(0)
STACK1 ENDS
CODE SEGMENT
  ASSUME CS:CODE,SS:STACK1,DS:DATA
START:MOV AX,DATA
      MOV DS,AX
      MOV CX,20
      LEA SI,BUF
AGAIN:MOV AL,[SI]
      CMP AL,5
      JLE NEXT1
      INC RES1
      INC SI
```

```
      DEC CX
      JNZ AGAIN
NEXT1:CMP AL,0
      JL NEXT2
      INC RES2
      INC SI
      DEC CX
      JNZ AGAIN
NEXT2:INC RES3
      INC SI
      DEC CX
      JNZ AGAIN
      MOV AH,2
      MOV BL,RES3
      ADD BL,30H
      MOV DL,BL
      INT 21H
      MOV AH,2
      MOV BL,RES2
      ADD BL,30H
      MOV DL,BL
      INT 21H
      MOV AL,RES1
      XOR AH,AH
      MOV BL,10
      MOV DL,AL
      OR  DL,30H      ;转换为ASCII码
      MOV AL,AH
      MOV AH,02H      ;显示十位数
      INT 21H
      MOV DL,AL
      OR  DL,30H
      MOV AH,02H      ;显示个位数
      INT 21H
      MOV AH,4CH
      INT 21H
CODE ENDS
      END START
```

（简单）计算表达式: Z=(5X+2Y-7)/2

设X、Y的值放在字节变量VARX、VARY中，结果存放在字节单元VARZ。

```
DATA SEGMENT
    VARX DB 10
    VARY DB 20
    VARZ DB ?
DATA ENDS
SSEG SEGMENT STACK
    DB 100H DUP(0)
SSEG ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA,SS:SSEG
START:
    MOV AX, DATA
    MOV DS, AX
```

```
    XOR AH, AH
    MOV AL, VARX
    MOV BL, AL
    MOV CL, 2
    SAL BL, CL
    ADD AL, BL
    XOR BH, BH
    MOV BL, VARY
    SAL BL, 1
    ADD AL, BL
    SUB AL, 7
    SAR AL, 1
    MOV VARZ, AL
    MOV AH, 4CH
    INT 21H
CODE ENDS
    END START
```

（复杂）从键盘输入两个2位十进制正数转换成十六进制数后，对这两个数相加，结果以十六进制数形式显示在屏幕上。

例如：

输入：12 34

屏幕上显示：2E

```
DSEG SEGMENT
    NUM1 DB 0   ;第一个十进制数的二进制存储变量
    NUM2 DB 0   ;第二个十进制数的二进制存储变量
DSEG ENDS

SSEG SEGMENT PARA STACK
    DB 20H DUP(0)
SSEG    ENDS

CSEG SEGMENT
    ASSUME CS:CSEG, DS:DSEG, SS:SSEG
MAIN    PROC FAR
START:
    MOV AX, DSEG
    MOV DS, AX
    CALL TOBIN  ;第一个十进制数输入并转换为二进制，结果存放在DL中
    MOV NUM1, DL
PUSH AX ;保护AX内容
MOV AH, 1
INT 21H ;输入空格过滤掉
POP AX  ;还原AX内容

CALL TOBIN ;第二个十进制数输入并转换为二进制，结果存放在DL中
MOV NUM2, DL

MOV BL, NUM1
ADD BL, NUM2    ;两个二进制数相加，结果送BL
```

```
CALL TOHEX  ;转化为16进制数输出
MOV AH, 4CH ;返回DOS系统
INT 21H
MAIN    ENDP

TOBIN PROC  ;结果放入DL寄存器
    PUSH AX ;保护寄存器内容
    PUSH BX
    MOV DL, 0
MOV AH, 1   ;输入十位ASCLL码
INT 21H
MOV AH, 0
SUB AL, 30H ; 转换为数字
MOV BL, 10  ;X10
MUL BL
MOV DL, AL

MOV AH, 1   ;输入个位ASCLL码
INT 21H
SUB AL, 30H ; 转换为数字
ADD DL, AL  ;求和放入DL

POP BX  ;还原寄存器内容
POP AX
RET
TOBIN ENDP

TOHEX PROC  ;取BL寄存器操作数进行转化,因为两位数相加99+99<255故最多八位二进制数
    PUSH AX
    PUSH CX
    PUSH DX
TENS:
    MOV CL, 4   ;存取移位次数
    MOV DL, 0F0H
    AND DL, BL  ;取BL的高四位到DL寄存器高四位
    SHR DL, CL  ;DL操作数除以16
    CMP DL, 0AH ;与10比较大小判断是否为字母
    JAE LETTER1 ;若是则跳转到字母输出
    ADD DL, 30H ;转化为对应ASCLL码
    MOV AH, 2   ;输出显示
    INT 21H
    JMP UNITS
LETTER1:
    ADD DL, 37H ;若为字母转换为ASCLL码要加37H
    MOV AH, 2   ;输出显示
    INT 21H
UNITS:
    XOR DL, DL  ;寄存器清零
    MOV DL, 0FH
    AND DL, BL  ;取BL第四位到DL寄存器低四位
    CMP DL, 0AH ;与10比较大小判断是否为字母
    JAE LETTER2
    ADD DL, 30H
    MOV AH, 2
    INT 21H
    JMP OVER
LETTER2:
```

```
    ADD DL, 37H
    MOV AH, 2
    INT 21H
OVER:
    POP DX
    POP CX
    POP AX
    RET
TOHEX ENDP

CSEG ENDS
    END START
```

总结

除了最后一个之外，其余的代码能手写出来我觉得就差不多了，毕竟也不是专业学这个的。

实在不会的，就记住我写的模板，至少套点分数上。

如果你能看到这，希望你对汇编的语句能有足够的掌握，不够的话还有时间继续复习，考试一定顺利通过！！

以下是汇编指令的知识点，建议可见我的博客，下面这篇是详细版本。 https://blog.csdn.net/Trae1ounG/article/details/129107484?spm=1001.2014.3001.5502

以及我的Github仓库 https://github.com/Trae1ounG/jisuanjizuchengyuanli

欢迎star或收藏本文！！！