目录

[《汇编语言》课后检测题 参考答案 1](#_Toc201490879)

[第1章 基础知识 1](#_Toc201490880)

[检测点1.1 1](#_Toc201490881)

[第2章 寄存器 1](#_Toc201490882)

[检测点2.1 1](#_Toc201490883)

[检测点2.2 2](#_Toc201490884)

[检测点2.3 3](#_Toc201490885)

[第3章 寄存器（内存访问） 3](#_Toc201490886)

[检测点 3.1 3](#_Toc201490887)

[检测点 3.2 5](#_Toc201490888)

[第4章 第一个程序 6](#_Toc201490889)

[第5章 [bx]和loop指令 6](#_Toc201490890)

[第6章 包含多个段的程序 6](#_Toc201490891)

[检测点6.1 6](#_Toc201490892)

[第7章 更灵活的定位内存地址的方法 8](#_Toc201490893)

[第8章 数据处理的两个基本问题 8](#_Toc201490894)

[第9章 转移指令的原理 8](#_Toc201490895)

[检测点9.1 8](#_Toc201490896)

[检测点9.2 11](#_Toc201490897)

[检测点9.3 12](#_Toc201490898)

[第10章 CALL和RET指令 13](#_Toc201490899)

[检测点10.1 13](#_Toc201490900)

[检测点10.2 13](#_Toc201490901)

[检测点10.3 13](#_Toc201490902)

[检测点10.4 14](#_Toc201490903)

[检测点10.5 14](#_Toc201490904)

[第11章 标志寄存器 15](#_Toc201490905)

[检测点 11.1 15](#_Toc201490906)

[检测点 11.2 16](#_Toc201490907)

[检测点 11.3 16](#_Toc201490908)

[检测点 11.4 18](#_Toc201490909)

《汇编语言》课后检测题 参考答案

# 基础知识

## 检测点1.1

（1）1个CPU的寻址能力为8KB，那么它的地址总线的宽度为 13 。2^13=8192B

（2）1KB的存储器有1024个存储单元。存储单元的编号 0 到 1023 。

（3）1KB的存储器可以存储 2^13 个bit， 2^10 个byte 。2^10=1024B =8192b

（4）1GB、1MB、1KB分别是2^30 、2^20 、2^10 Byte 。

（5）8080、8088、80286、80386的地址总线宽度分别为16根、20根、24根、32根，则它们的寻址能力分别为： 2^6 （KB）、2^0 （MB）、2^4 （MB）、2^2 （GB）

（6）8080、8088、8086、80286、80386的数据总线宽度分别为8根、8根、16根、16根、32根。则它们一次可以传送的数据为： 1 （B）、 1 （B）、 2 （B） 、 2 （B）、 4 （B）。

（7）从内存中读取1024字节的数据，8086至少要读 512 次（16位，一次传送2字节），80386至少要读 256 次（32位，一次传送4字节）。

（8）在存储器中，数据和程序以 二进制 形式存放。

# 寄存器

## 检测点2.1

**（1）写出每条汇编指令执行后相关寄存器中的值。**

mov ax,62627 AX=F4A3H

mov ah,31H AX=31A3H

mov al,23H AX=3123H

add ax,ax AX=6246H

mov bx,826CH BX=826CH

mov cx,ax CX=6246H

mov ax,bx AX=826CH

add ax,bx AX=04D8H

mov al,bh AX=0482H

mov ah,bl AX=6C82H

add ah,ah AX=D882H

add al,6 AX=D888H

add al,al AX=D810H

mov ax,cx AX=6246H

（2）只能使用目前学过的汇编指令，最多使用4条指令，编程计算2的4次方。

mov ax,2

add ax,ax

add ax,ax

add ax,ax

## 检测点2.2

（1）给定段地址为0001H，仅通过变化偏移地址寻址，CPU的寻址范围\_\_00010H\_\_\_\_\_到\_\_\_1000FH\_\_\_\_。

（2）有一数据存放在内存20000H单元中，现给定段地址为SA，若想用偏移地址寻到此单元。则SA应满足的条件是：最小为\_\_1001H\_\_，最大为\_2000H\_。

## 检测点2.3

下面3条指令执行后，CPU修改几次IP？都是在什么时候？最后IP中的值是多少？

1. mov ax, bx
2. sub ax, ax
3. jmp ax

共修改4次；

第一次变化，输入输出控制电路将mov ax, bx对应的机器指令送入指令缓冲器后，IP中的值自动增加。

第二次变化，输入输出控制电路将sub ax, bx对应的机器指令送入指令缓冲器后，IP中的值自动增加。

第三次变化，输入输出控制电路将jmp ax对应的机器指令送入指令缓冲器后，IP中的值自动增加。

第四次变化，执行控制器执行第3条汇编指令对应的机器指令后，指令指针寄存器IP中的数据变为寄存器ax中的数据。

最后IP中的值为0000H

# 寄存器（内存访问）

## 检测点 3.1

（1）使用 d 指令查看 0000:0000~0000:001F 的内存值，并写出每条汇编指令执行完后相关寄存器中的值。

0000:0000 70 80 F0 30 EF 60 30 E2-00 80 80 12 66 20 22 60

0000:0010 62 26 E6 D6 CC 2E 3C 3B-AB BA 00 00 26 06 66 88

每条汇编执行完成后相关寄存器中的值为：

|  |  |  |
| --- | --- | --- |
| 指令 | AX | BX |
| mov ax,1 | 0001 | 0000 |
| mov ds,ax | 0001 | 0000 |
| mov ax,[0000] | 2662 | 0000 |
| mov bx,[0001] | 2662 | E626 |
| mov ax,bx | E626 | E626 |
| mov ax,[0000] | 2662 | E626 |
| mov bx,[0002] | 2662 | D6E6 |
| add ax,bx | FD48 | D6E6 |
| add ax,[0004] | 2C14 | D6E6 |
| mov ax,0 | 0000 | D6E6 |
| mov al,[0002] | 00E6 | D6E6 |
| mov bx,0 | 00E6 | 0000 |
| mov bl,[000C] | 00E6 | 0026 |
| add al,bl | 000C | 0026 |

（2）内存中的情况如下图：

各寄存器的初值为：CS=2000H、IP=0、DS=1000H、AX=0、BX=0。

① 使用汇编代码写出 CPU 执行的指令序列。

② 写出 CPU 每条指令执行后，相关寄存器中的数值。

③ 程序和数据的区别。

CPU 从 CS:IP 处开始执行指令，执行完每条指令后，IP 的值加上上一条指令的长度。

使用 jmp 指令修改 CS:IP 的值，jmp 段地址:偏移地址，如 jmp 2AE3:3 后，CS=2AE3H、IP=0003H，并且 CPU 将从 2AE3:3 处读取指令并执行；或 jmp 某一合法寄存器名，如 jmp AX 后，IP=AX。

CPU 总是从 CS(2000H):IP(0) 处开始执行指令：

指令 AX BX CS IP DS

mov ax,6622H 6622 0000 2000 0003 1000

jmp 0ff0:0100H 6622 0000 1000 0000 1000

mov ax,2000H 2000 0000 1000 0003 1000

mov ds,ax 2000 0000 1000 0005 2000

mov ax,[0008] C189 0000 1000 0008 2000

mov ax,[0002] EA66 0000 1000 000B 2000

## 检测点 3.2

（1）补全程序，使其可以将 10000H~1000FH 中的 8 个字逆序拷贝到 20000H~2000FH 中。

mov ax,1000H

mov ds,ax

mov ax,2000H ;

mov ss,ax;

mov sp,0010H;

push [0]

push [2]

push [4]

push [6]

push [8]

push [A]

push [C]

push [E]

（2）补全程序，使其可以将 10000H~1000FH 中的 8 个字逆序拷贝到 20000H~2000FH 中。

mov ax,2000H

mov ds,ax

mov ax,1000H

mov ss,ax

mov sp,0000H

pop [E]

pop [C]

pop [A]

pop [8]

pop [6]

pop [4]

pop [2]

pop [0]

# 第一个程序

# [bx]和loop指令

# 包含多个段的程序

## 检测点6.1

（1）下面的程序实现依次用内存 0:0~0:15 单元中的内容改写程序中的数据，完成程序：

assume cs:codesg

codesg segment

dw 0123H,0456H,0789H,0abcH,0defH,0fedH,0cbaH,0987H

start: mov ax,0

mov ds,ax

mov bx,0

mov cx,8

s: mov ax,[bx]

mov cs:[bx],ax

add bx,2

loop s

mov ax,4c00h

int 21h

codesg ends

end start

（2）下面的程序实现依次用内存 0:0~0:15 单元中的内容改写程序中的数据，数据的传送用栈进行

assume cs:codesg

codesg segment

dw 0123H,0456H,0789H,0abcH,0defH,0fedH,0cbaH,0987H

dw 0,0,0,0,0,0,0,0,0,0

start: mov ax,\_\_ ;mov ax,cs

mov ss,ax

mov sp,\_\_ ;mov sp,36，即将 0:10h~0:24h 部分当作栈区使用

mov ax,0

mov ds,ax

mov bx,0

mov cx,8

s: push [bx]

pop cs:[bx]

add bx,2

loop s

mov ax,4c00h

int 21h

codesg ends

end start

# 更灵活的定位内存地址的方法

# 数据处理的两个基本问题

# 转移指令的原理

## 检测点9.1

1. 程序如下：

assume cs:code

data segment

?

data ends

code segment

start:

mov ax,data

mov ds,ax ;寄存器DS指向数据段data

mov bx,0

jmp word ptr [bx+1] ;段内转移，偏移地址由[bx+1]给出

code ends

end start

若要使程序中的 jmp 指令执行后，CS:IP 指向程序的第一条指令，在 data 段中应该定义哪些数据？

jmp word ptr [bx+1] 实现段内转移，且 jmp 指令执行后 CS 不变、(IP)=ds:[bx+1]。

由于第一条指令的偏移地址为 0，要使 CS:IP 指向第一条指令，则 ds:[bx+1] 的值为零，而 ds:[bx+1] 表示 data 段的第二个字节（mov bx,0）。所以在 data 段中将第三、四个字节定义为零即可（不唯一）：

data segment

db 4 dup (0) ;使用dup指令定义4个字节，值都是0

data ends

（2）程序如下：

assume cs:code

data segment

dd 12345678H

data ends

code segment

start:

mov ax,data

mov ds,ax

mov bx,0

mov [bx],\_\_\_\_

mov [bx+2],\_\_\_\_

jmp dword ptr ds:[0]

mov ax,4c00h

int 21h

code ends

end start

补全程序，使 jmp 指令执行后，CS:IP 指向程序的第一条指令。

jmp dword ptr ds:[0] 实现段间转移，目的段地址由内存单元高地址给出、偏移地址由内存单元低地址给出。

程序中内存单元的值是 ds:[0]，结合数据段的内容为 12345678h。所以，jmp 执行后，目的段地址（CS）为 5678h、偏移地址（IP）为 1234h。显然，待填空部分操作的是数据段的内存单元，即改变数据段的内容使 jmp 执行后 CS:IP 指向程序的第一条指令。

覆盖数据段的内容，高地址（CS）为寄存器 CS 的值、低地址（IP）为 0。代码段部分为：

code segment

start:

mov ax,data

mov ds,ax

mov bx,0

mov [bx],bx ;将寄存器BX的内容赋值给数据段中的低地址

mov [bx+2],cs ;将寄存器CS的内容赋值给数据段中的高地址

jmp dword ptr ds:[0]

mov ax,4c00h

int 21h

code ends

（3）用 Debug 查看内存，结果如下：

2000:1000 BE 00 06 00 00 00 ......

则此时，CPU 执行指令：

mov ax,2000H

mov es,ax

jmp dword ptr es:[1000H]

后，寄存器 CS 和寄存器 IP 的值为多少？

jmp dword ptr es:[1000H] 实现段间转移，目的段地址由内存单元高地址给出、偏移地址由内存单元低地址给出。

2000:[1000H] 所指内存单元的高地址值为 0006，低地址值为 00BE。所以，(CS)=0006、(IP)=00BE。

## 检测点9.2

补全编程，利用 jcxz 指令，实现在内存 2000H 段中查找第一个值为 0 的字节，找到后，将它的偏移地址存储在 dx 中。

code segment

start:

mov ax,2000H

mov ds,ax

mov bx,0

s: mov cl,ds:[bx] ;将内存单元的值依次赋值给寄存器CL

mov ch,0 ;将CH填充为0，和CL组合判断CX是否为零

jcxz ok ;如果(CX)等于零则跳转到标号ok处，即结束查找

add bx,1 ;每次向后偏移一个字节

jmp short s

ok: mov dx,bx ;偏移地址存储在寄存器DX中

mov ax,4c00h

int 21h

code ends

## 检测点9.3

补全编程，利用 loop 指令，实现在内存 2000H 段中查找第一个值为 0 的 byte（字节），找到后，将它的偏移地址存储在 dx 中。

code segment

start:

mov ax,2000H

mov ds,ax

mov bx,0

s: mov cl,[bx]

mov ch,0

add cx,1

inc bx ;寄存器BX加1表示偏移1个字节查找下一个字节

loop s

ok: dec bx ;dec与inc指令的功能相反，将对应寄存器/内存单元的值减1

mov dx,bx

mov ax,4c00h

int 21h

code ends

# CALL和RET指令

## 检测点10.1

补全程序，实现从内存1000：0000处开始执行指令。

assume cs:code

stack segment

db 16 dup (0)

stack ends

code segment

start: mov ax,stack

mov ss,ax

mov sp,16

mov ax, 1000h

push ax

mov ax, 0

push ax

retf

code ends

end start

## 检测点10.2

下面的程序执行后，ax中的数值为多少？

内存地址 机器码 汇编指令 执行后情况

1000:0 b8 00 00 mov ax,0 ax=0 ip指向1000:3

1000:3 e8 01 00 call s pop ip ip指向1000:7

1000:6 40 inc ax

1000:7 58 s:pop ax ax=6

用debug进行跟踪确认，“call 标号”是将该指令后的第一个字节偏移地址入栈，再转到标号处执行指令。

## 检测点10.3

下面的程序执行后，ax中的数值为多少？

内存地址 机器码 汇编指令 执行后情况

1000:0 b8 00 00 mov ax,0 ax=0,ip指向1000:3

1000:3 9a 09 00 00 10 call far ptr s pop cs,pop ip,ip指向1000:9

1000:8 40 inc ax

1000:9 58 s:pop ax ax=8h

add ax,ax ax=10h

pop bx bx=1000h

add ax,bx ax=1010h

## 检测点10.4

下面的程序执行后，ax中的数值为多少？

内存地址 机器码 汇编指令 执行后情况

1000:0 b8 06 00 mov ax,6 ax=6,ip指向1000:3

1000:3 ff d0 call ax pop ip,ip指向1000:6

1000:5 40 inc ax

1000:6 58 mov bp,sp bp=sp=fffeh

add ax,[bp] ax=[6+ds:(fffeh)]=6+5=0bh

用debug进行跟踪确认，“call ax(16位reg)”是先将该指令后的第一个字节偏移地址ip入栈，再转到偏移地址为ax(16位reg)处执行指令。

## 检测点10.5

（1）下面的程序执行后，ax中的数值为多少？

assume cs:code

stack segment

dw 8 dup (0)

stack ends

code segment

start: mov ax,stack

mov ss,ax

mov sp,16

mov ds,ax

mov ax,0

call word ptr ds:[0eh]

inc ax

inc ax

inc ax

mov ax,4c00h

int 21h

code ends

end start

推算：

执行call word ptr ds:[0eh]指令时，先cs入栈，再ip=11入栈，最后ip转移到(ds:[0eh])。(ds:[0eh])=11h，执行inc ax……最终ax=3

（2）下面的程序执行后，ax和bx中的数值为多少？

assume cs:codesg

stack segment

dw 8 dup(0)

stack ends

codesg segment

start:

mov ax,stack

mov ss,ax

mov sp,10h

mov word ptr ss:[0],offset s ;(ss:[0])=1ah

mov ss:[2],cs ;(ss:[2])=cs

call dword ptr ss:[0] ;cs入栈,ip=19h入栈,转到cs:1ah处执行指令

;(ss:[4])=cs,(ss:[6])=ip

nop

s: mov ax,offset s ;ax=1ah

sub ax,ss:[0ch] ;**ax**=1ah-(ss:[0ch])=1ah-19h**=1**

mov bx,cs ;bx=cs＝0c5bh

sub bx,ss:[0eh] ;**bx**=cs-cs**=0**

mov ax,4c00h

int 21h

codesg ends

end start

# 标志寄存器

## 检测点 11.1

写出下面每条指令执行后，ZF、PF、SF 等标志位的值。

指令 ZF PF SF

sub al,al 1 1 0

mov al,1 1 1 0

push ax 1 1 0

pop bx 1 1 0

add al,bl 0 0 0

add al,10 0 1 0

mul al 0 1 0

## 检测点 11.2

写出下面每条指令后，ZF、PF、SF、CF、OF 等标志位的值。

指令 CF OF SF ZF PF

sub al,al 0 0 0 1 1

mov al,10H 0 0 0 1 1

add al,90H 0 0 1 0 1

mov al,80H 0 0 1 0 1

add al,80H 1 1 0 1 1

mov al,0FCH 1 1 0 1 1

add al,05H 1 0 0 0 0

mov al,7DH 1 0 0 0 0

add al,0BH 0 1 1 0 1

## 检测点 11.3

（1）补全下面的程序，统计 F000:0 处 32 个字节中，大小在 [32, 128] 的数据的个数。

mov ax,0f000h

mov ds,ax

mov bx,0

mov dx,0

mov cx,32

s:

mov al,[bx]

cmp al,32

\_jb s0\_\_\_\_\_\_\_\_\_\_

cmp al,128

\_ja\_s0\_\_\_\_\_\_\_\_\_

inc dx

s0:

inc bx

loop s

（2）补全下面的程序，统计 F000:0 处 32 个字节中，大小在 (32, 128) 的数据的个数。

mov ax,0f000h

mov ds,ax

mov bx,0

mov dx,0

mov cx,32

s:

mov al,[bx]

cmp al,32

jna s0

cmp al,128

jnb s0

inc dx

s0:

inc bx

loop s

## 检测点 11.4

下面的程序执行后：(ax)=？ (AX)=0045H

mov ax,0

push ax

popf

mov ax,0fff0h

add ax,0010h

pushf

pop ax

and al,11000101B

and ah,00001000B