**实验二 分支与循环程序设计**

班级： 23计科（）班 姓名： 学号：

**一、实验目的**

1、掌握程序设计中的3种基本结构（顺序结构、选择程序、循环程序）。

2、熟练使用汇编语言的指令：数据传送类指令、数据运算类指令、逻辑判断类指令与转移指令、循环指令等。

3、初步了解系统功能调用的使用方法，尝试使用01H号功能调用进行字符输入的方法及使用02H号功能调用进行字符输出（显示）的方法。

**二、实验环境与设备**

计算机一台，并且安装Windows操作系统和模拟器软件EMU8086或DOSBOX。

**三、预备知识**

指令INT 21H 会调用DOS操作系统提供的中断类型码为21H的系统功能调用，其中包含了DOS提供给程序员在编程时可使用的常见功能，如键盘输入、显示器输出、打印输出、串口通信、文件操作、日期操作等功能。可以在程序中，通过INT指令调用这些功能，类似于在C程序中调用库函数，只不过两者在调用方法不同。

指令INT 21H的用法：

准备调用参数（通常是相关寄存器赋值）

MOV AH, 功能号

INT 21H

DOS系统功能调INT 21H的具体用法如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| ***AH*** | **功能** | **调用参数** | **返回参数** |
| *00* | 程序终止(同INT 20H) | CS=程序段前缀 |  |
| *01* | 键盘输入并回显 |  | AL=输入字符 |
| *02* | 显示输出 | DL=输出字符 |  |
| *03* | 异步通迅输入 |  | AL=输入数据 |
| *04* | 异步通迅输出 | DL=输出数据 |  |
| *05* | 打印机输出 | DL=输出字符 |  |
| *06* | 直接控制台I/O | DL=FF(输入)  DL=字符(输出) | AL=输入字符 |
| *07* | 键盘输入(无回显) |  | AL=输入字符 |
| *08* | 键盘输入(无回显)  检测Ctrl-Break |  | AL=输入字符 |
| *09* | 显示字符串 | DS:DX=以'$'结束的字符串的首地址 |  |
| *0A* | 键盘输入到缓冲区 | DS:DX=缓冲区首地址  (DS:DX)=缓冲区最大字符数 | (DS:DX+1)=实际输入的字符数 |
| *0B* | 检验键盘状态 |  | AL=00 有输入  AL=FF 无输入 |
| *0C* | 清除输入缓冲区并  请求指定的输入功能 | AL=输入功能号  (1,6,7,8,A) |  |
| *0D* | 磁盘复位 |  | 清除文件缓冲区 |
| *0E* | 指定当前缺省的磁盘驱动器 | DL=驱动器号 0=A,1=B,... | AL=驱动器数 |
| *0F* | 打开文件 | DS:DX=FCB首地址 | AL=00 文件找到  AL=FF 文件未找到 |
| *10* | 关闭文件 | DS:DX=FCB首地址 | AL=00 目录修改成功  AL=FF 目录中未找到文件 |
| *11* | 查找第一个目录项 | DS:DX=FCB首地址 | AL=00 找到  AL=FF 未找到 |
| *12* | 查找下一个目录项 | DS:DX=FCB首地址  (文件中带有\*或?) | AL=00 找到  AL=FF 未找到 |
| *13* | 删除文件 | DS:DX=FCB首地址 | AL=00 删除成功  AL=FF 未找到 |
| *14* | 顺序读 | DS:DX=FCB首地址 | AL=00 读成功  =01 文件结束,记录中无数据  =02 DTA空间不够  =03 文件结束,记录不完整 |
| *15* | 顺序写 | DS:DX=FCB首地址 | AL=00 写成功  =01 盘满  =02 DTA空间不够 |
| *16* | 建文件 | DS:DX=FCB首地址 | AL=00 建立成功  =FF 无磁盘空间 |
| *17* | 文件改名 | DS:DX=FCB首地址  (DS:DX+1)=旧文件名  (DS:DX+17)=新文件名 | AL=00 成功  AL=FF 未成功 |
| *19* | 取当前缺省磁盘驱动器 |  | AL=缺省的驱动器号 0=A, 1=B, 2=C, ... |
| *1A* | 置DTA地址 | DS:DX=DTA地址 |  |
| *1B* | 取缺省驱动器FAT信息 |  | AL=每簇的扇区数  DS:BX=FAT标识字节  CX=物理扇区大小  DX=缺省驱动器的簇数 |
| *1C* | 取任一驱动器FAT信息 | DL=驱动器号 | 同上 |
| *21* | 随机读 | DS:DX=FCB首地址 | AL=00 读成功  =01 文件结束  =02 缓冲区溢出  =03 缓冲区不满 |
| *22* | 随机写 | DS:DX=FCB首地址 | AL=00 写成功  =01 盘满  =02 缓冲区溢出 |
| *23* | 测定文件大小 | DS:DX=FCB首地址 | AL=00 成功(文件长度填入FCB)  AL=FF 未找到 |
| *24* | 设置随机记录号 | DS:DX=FCB首地址 |  |
| *25* | 设置中断向量 | DS:DX=中断向量  AL=中断类型号 |  |
| *26* | 建立程序段前缀 | DX=新的程序段前缀 |  |
| *27* | 随机分块读 | DS:DX=FCB首地址  CX=记录数 | AL=00 读成功  =01 文件结束  =02 缓冲区太小,传输结束  =03 缓冲区不满 |
| *28* | 随机分块写 | DS:DX=FCB首地址  CX=记录数 | AL=00 写成功  =01 盘满  =02 缓冲区溢出 |
| *29* | 分析文件名 | ES:DI=FCB首地址  DS:SI=ASCIIZ串  AL=控制分析标志 | AL=00 标准文件  =01 多义文件  =02 非法盘符 |
| *2A* | 取日期 |  | CX=年  DH:DL=月:日(二进制) |
| *2B* | 设置日期 | CX:DH:DL=年:月:日 | AL=00 成功  =FF 无效 |
| *2C* | 取时间 |  | CH:CL=时:分  DH:DL=秒:1/100秒 |
| *2D* | 设置时间 | CH:CL=时:分  DH:DL=秒:1/100秒 | AL=00 成功  =FF 无效 |
| *2E* | 置磁盘自动读写标志 | AL=00 关闭标志  AL=01 打开标志 |  |
| *2F* | 取磁盘缓冲区的首址 |  | ES:BX=缓冲区首址 |

常用子功能用法举例：

1. 功能号是01H，键盘输入并回显，调用参数：无，返回参数：AL=输入字符。

|  |
| --- |
| MOV AH, 1 ; AH存入子功能号1  INT 21H ; 调用21H中断，等待用户键盘输入一个字符，并存入AL |

2. 功能号是02H，显示输出，调用参数：DL=输出字符，返回参数：无。

|  |
| --- |
| MOV AH, 2 ; AH存入子功能号2  MOV DL, ‘A’ ; DL存入要显示的字符‘A’  INT 21H ; 调用21H中断，屏幕上显示DL中的字符 |

3. 功能号是09H，显示字符串，调用参数：DS:DX=以'$'结束的字符串的首地址，返回参数：无。

|  |
| --- |
| 数据段：  定义变量，并初始化为字符串：  STR DB “hello world”, ‘$’  代码段：  MOV AH, 9 ; AH存入子功能号9  MOV DX, OFFSET STR ; DX中存入字符串的首地址  INT 21H ; 调用21H中断，屏幕上显示DX指向的字符串“hello world” |

**四、实验内容**

* + 1. 计算1+2+3+…+100，并利用21H中断的02H或09号功能调用将十进制结果显示在屏幕上。
    2. 利用21H中断的01H号功能调用输入10个一位数字，将其由数字字符转换为相应整数，并依次保存到字节变量BUF。编程求出这10个数中的最大数和最小数，分别存入字节变量MAX和MIN，并分别将其在屏幕上显示出来。
    3. 输入一个不大于65535的十进制非负整数，判断其是否为素数，如果是素数，输出字符串“It’s a prime.”，否则输出字符串“It’s not a prime.”。

**五、实验要求**

1、上机前要作好充分准备，包括程序流程设计（用程序流程图、C程序等任一形式描述）、源程序清单、测试方法等。

2、要熟悉与实验有关的软件(如编辑程序、汇编程序、连接程序和调试程序等)的使用方法。在程序的调试过程中，有意识地学习及掌握debug程序的各种操作命令，以便掌握程序的调试方法及技巧。

3、程序调试完后，须由实验辅导教师在机器上检查运行结果，经教师认可后的源程序方可通过打印输出。每个实验完成后，应写出实验报告。实验报告的要求如下：

(1)实验目的：对本次实验的目的加以说明。

(2)实验内容：本次实验你所完成的具体内容加以说明。

(3)设计思想：绘制程序流程图或类C伪代码，说明程序算法、数据结构、主要符号名等。

(4)程序代码：经辅导教师确认后的程序清单。

(5)结果分析：包括调试情况，如上机时遇到的问题及解决办法、观察到的现象及其分析．对程序设计技巧的总结及分析；程序的输出结果及对结果的分析；实验的心得体会等。

实验报告

班级： 23计科（）班 姓名： 盲灯 学号：

一、实验目的：

1、掌握程序设计中的3种基本结构（顺序结构、选择程序、循环程序）。

2、熟练使用汇编语言的指令：数据传送类指令、数据运算类指令、逻辑判断类指令与转移指令、循环指令等。

3、初步了解系统功能调用的使用方法，尝试使用01H号功能调用进行字符输入的方法及使用02H号功能调用进行字符输出（显示）的方法。

二、实验内容：

* + 1. 计算1+2+3+…+100，并利用21H中断的02H或09号功能调用将十进制结果显示在屏幕上。

程序从 main proc 入口开始执行，分为以下几个阶段：

（1）数据段初始化

mov ax, data

mov ds, ax

通过将 data 段地址加载至 DS 寄存器，实现对数据段中字符串 prompt 的访问。

（2）循环求和，变量 sum 与 i

mov ax, 0 ; sum ← 0

mov si, 1 ; i ← 1

利用 AX 保存累加结果，SI 为循环计数器，进入循环计算 1~99 的总和。

（3）将十进制结果逐位提取，逆序入栈

mov bx, 10

cwd

idiv bx

add dl, '0'

push dx

使用 除 10 取余法 将整数转换为十进制字符，存入栈中，实现了数字字符的反序排列。

（4）从栈中弹出字符并依次输出

pop dx

mov ah, 2

int 21h

通过 int 21h / ah = 2 将每个字符输出到屏幕，实现 高位数字先输出。

（5）输出换行与提示信息

mov dl, 13 / 10

mov ah, 2

int 21h

添加换行，然后调用 int 21h / ah = 9 显示提示字符串 prompt。

（6）等待任意键与程序退出

mov ah, 1

int 21h ; 等待键盘输入

mov ah, 4ch

int 21h ; 正常退出程序

通过 DOS 功能调用等待用户操作并正常退出返回操作系统。

* + 1. 利用21H中断的01H号功能调用输入10个一位数字，将其由数字字符转换为相应整数，并依次保存到字节变量BUF。编程求出这10个数中的最大数和最小数，分别存入字节变量MAX和MIN，并分别将其在屏幕上显示出来。

程序从 main proc 入口开始执行，整体分为以下几个阶段：

（1）数据段初始化

mov ax, data

mov ds, ax

通过将 data 段地址加载至 DS 寄存器，实现对变量 buf、max、min 及提示字符串 prompt 的访问，为后续输入输出与变量操作做好准备。

（2）循环输入数字，存入数组 buf

mov si, 0

; loop1: cmp si, 10

; jne ...

使用 SI 作为循环变量 i，通过键盘输入中断 int 21h / ah = 1 逐个获取用户输入的 10 个一位数字字符。使用 cmp al, '0' 与 cmp al, '9' 判断字符合法性，并通过 sub al, '0' 转为数值，存入 buf[si]。

（3）首次输入时初始化最大值与最小值

cmp si, 0

jne less\_than\_min

mov max, al

mov min, al

在 i == 0 时，将第一个输入的数字初始化为最大值和最小值，为后续比较提供参考基准。

（4）比较并更新最值：min 与 max

cmp al, min

jb mov min, al

cmp al, max

ja mov max, al

对每个输入的数字与当前 min 和 max 进行比较，并在发现更小或更大值时立即更新。实现实时维护当前最小值与最大值。

（5）输出最大值与最小值

mov ah, 9

mov dx, offset string1

int 21h ; 输出 "The maximum value is"

mov dl, max

add dl, '0'

int 21h ; 输出最大值字符

mov ah, 9

mov dx, offset string2

int 21h ; 输出 "The minimum value is"

mov dl, min

add dl, '0'

int 21h ; 输出最小值字符

使用 DOS 字符串输出功能（int 21h / ah=9）和单字符输出功能（int 21h / ah=2）将提示语与最值内容依次显示。

（6）输出换行与提示信息

mov dl, 13

int 21h

mov dl, 10

int 21h

mov dx, offset prompt

mov ah, 9

int 21h

先输出换行符 CR/LF，再通过 int 21h / ah = 9 输出提示信息 "press any key to continue..."。

（7）等待按键与程序退出

mov ah, 1

int 21h ; 等待按键

mov ah, 4Ch

mov al, 0

int 21h ; 正常退出程序

调用 int 21h / ah = 1 等待用户输入任意键后，使用 int 21h / ah = 4Ch 正常结束程序，并返回到 DOS 操作系统。

* + 1. 输入一个不大于65535的十进制非负整数，判断其是否为素数，如果是素数，输出字符串“It’s a prime.”，否则输出字符串“It’s not a prime.”。

程序从 main proc 入口开始执行，整体分为以下几个阶段：

（1）数据段初始化

mov ax, data

mov ds, ax

通过将数据段地址加载到 DS 寄存器，完成 prime\_msg、not\_prime\_msg 和 prompt 三个提示字符串的地址访问初始化，为后续输出提示信息提供数据支持。

（2）输入十进制整数并转换为数值

mov ah, 1

int 21h

程序通过循环调用 int 21h / ah = 1 从键盘读取每一个字符（数字），利用 ASCII 码范围 '0'~'9' 判断合法性，若合法则通过：

sub al, '0'

将字符转换为对应的十进制数值，并实时更新当前值 n 的累积，使用：

BX ← BX \* 10 + digit

的算法转换为整型存储在 BX 中，实现对不超过 65535 的多位十进制整数输入与解析。

（3）判断边界值 n ≤ 1 的特殊情况

cmp bx, 1

jbe s1

若用户输入的整数 n ≤ 1，则直接跳转至标签 s1，将素数标志 is\_prime（存放在 CX）置为 0，表示该数不是素数，跳过后续判断过程。

（4）枚举 i = 2 ~ √n，判断能否整除

mov si, 2

loop\_start:

mov ax, si

mul si ; 计算 i \* i

cmp ax, bx

ja s3 ; 若 i\*i > n，则跳出

程序使用 SI 作为试除因子 i，循环执行从 2 到 √n 的范围。若 i\*i > n，则说明已经无须再检查后续可能因子，跳出判断。

（5）判断 n % i 是否为 0

mov ax, bx

xor dx, dx

div si

cmp dx, 0

每轮循环中通过 DIV 指令进行除法操作，将 n 除以当前 i，若余数 dx == 0，则说明 n 可被整除，不是素数，此时将 CX 清零，表示 is\_prime = 0 并立即跳出。

（6）判断是否为素数并输出提示信息

cmp cx, 1

jne not\_prime

根据 CX 寄存器中标志位是否为 1 判断当前输入是否为素数。若是素数，则调用：

mov dx, offset prime\_msg

mov ah, 9

int 21h

输出 It's a prime. 字符串，否则输出 It's not a prime.。

（7）输出换行、提示信息并等待按键

mov ah, 9

mov dx, offset prompt

int 21h

mov ah, 1

int 21h

程序在输出结果后，调用字符串输出与字符输入中断，提示用户“press any key to continue...”，并等待任意按键按下。

（8）正常退出程序，返回操作系统

mov ah, 4Ch

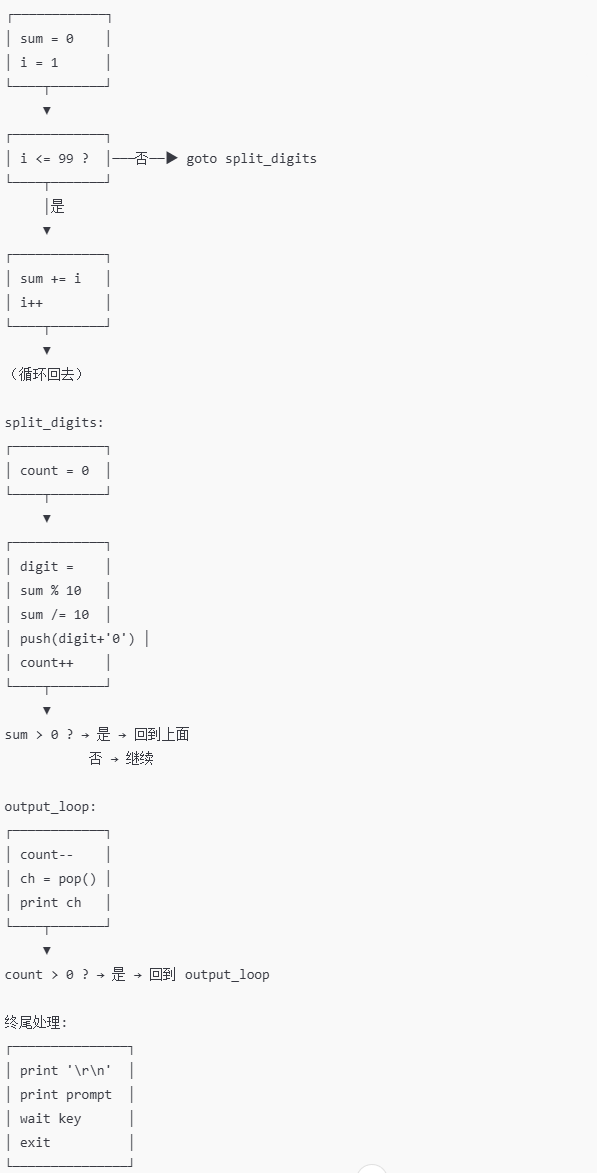
int 21h

调用 int 21h / ah = 4Ch 功能号，正常结束当前程序并返回到 DOS 操作系统，返回值为 0。

1. 设计思想：

1.计算1+2+3+…+100，并利用21H中断的02H或09号功能调用将十进制结果显示在屏幕

（1）.程序流程图：



（2）.C类伪代码：

*/\*计算1+2+3+…+100，并将结果显示在屏幕上。*

  在汇编语言中，21H中断中没有输入输出整数的功能，只能输入输出单个字符。

  为方便映射到汇编程序，此C程序中在输入输出时只能使用getchar、putchar函数。

\*/

#include <stdio.h>

int main(){

 short sum=0; *//累加和初始化*

 short i=1;  *//累加项初始化*

*//计算1+2+3+…+100*

 while(i<=100){

  sum+=i;

  i++;

 }

 char digits[5]; *//数组digits存放累加和sum整数分离出的各位数字字符*

 short count=0; *//digits中存入的数字字符的个数*

*//从累加和sum中逆序分离出各位数字，并存入数组digits*

 do{

  char digit=sum%10;

  digit+='0';

  digits[count]=digit;

  count++;

  sum/=10;

 }while(sum!=0);

*//逆序输出数组digits中的各个字符*

 do{

  count--;

  putchar(digits[count]);

 }while(count>0);

*//输出回车符*

 putchar('\n');

 return 0;

}

（3）.程序算法

程序的核心算法可以分为三段：

①. 求和算法

使用递增计数器 SI 从 1 到 99，将每一项加到累加器 AX 中，完成求和：

sum = 1 + 2 + ... + 99

②. 十进制分离与转字符算法

使用除10取余法提取 sum 的十进制各位数字，转换为字符后依次压栈，实现逆序存储。

③. 栈弹出并显示算法

利用 pop 指令按原始高位顺序将字符逐个输出到屏幕。

(4).数据结构说明:

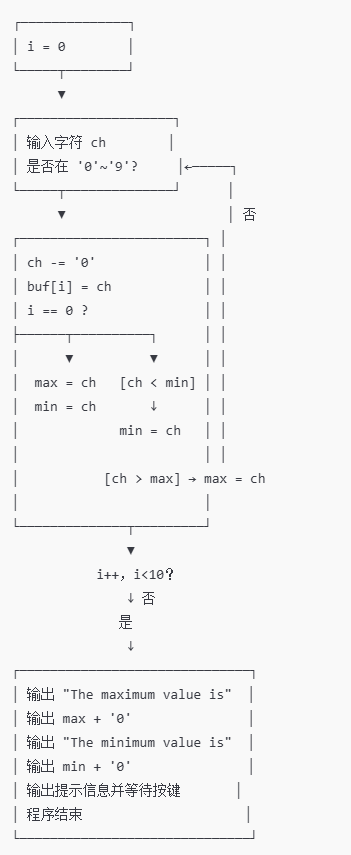
| 名称 | 类型 | 含义与作用 |
| --- | --- | --- |
| sum（AX） | 16位寄存器 | 用作累加器，保存 1 到 99 的求和结果 |
| i（SI） | 16位寄存器 | 作为循环变量（计数器） |
| prompt | 字符串常量 | 存储提示语，用于输出 |
| 栈 | 隐式结构 | 存储十进制数字字符，完成逆序输出 |

(5).主要符号名

| 符号名 | 类型/作用 | 说明 |
| --- | --- | --- |
| main | 主过程入口 | 相当于 C 的 int main() |
| prompt | 数据段字符串 | 提示信息 "press any key to continue..." |
| summation\_entrance | 循环入口标签 | 求和循环开始的位置 |
| summation\_exit | 循环出口标签 | 求和完成后跳出循环的位置 |
| split\_digit\_entrance | 拆分数字入口 | 用于从 sum 中依次提取十进制位 |
| output\_digit\_entrance | 输出字符入口 | 用于从栈中弹出数字并逐个输出 |

* 1. 利用21H中断的01H号功能调用输入10个一位数字，将其由数字字符转换为相应整数，并依次保存到字节变量BUF。编程求出这10个数中的最大数和最小数，分别存入字节变量MAX和MIN，并分别将其在屏幕上显示出来。

（1）.程序流程图：



（2）.C类伪代码：

*/\*输入10个一位数字，保存到数组buf，求最大数max和最小数min，并输出。*

  在汇编语言中，21H中断中没有输入输出整数的功能，只能输入输出单个字符。

  为方便映射到汇编程序，此C程序中在输入输出时只能使用getchar、putchar函数。

\*/

#include <stdio.h>

char buf[10];

char max;

char min;

int main(){

*//输入10个一位数，并求极值*

 short i=0;

 while(i<10){

  char ch;

  while((ch=getchar())<'0' || ch>'9');

  ch-='0';

  buf[i]=ch;

  if(i==0){

   max=ch;

   min=ch;

  }else if(ch<min){

   min=ch;

  }else if(ch>max){

   max=ch;

  }

  i++;

 }

*//输出极值*

 printf("\nThe maximum value is ");

 putchar(max+'0');

 printf("\nThe minimum value is ");

 putchar(min+'0');

 putchar('\n');

 return 0;

}

（3）程序算法

程序的核心算法可以分为三段：

①.输入与存储算法  
使用递增计数器 SI 控制输入次数，通过循环读取用户输入的 10 个一位数字字符，并减去 '0' 转换为数值后存入数组 buf[10] 中。

②.最大最小值更新算法  
输入第一个数时同时初始化 max 和 min，之后每次输入新值都与当前 max、min 比较，若更大或更小则更新，保证实时维护当前极值。

③.输出算法  
使用 int 21h 的子功能 AH=9 和 AH=2 输出提示语与极值字符，并等待用户输入任意键后退出程序。

(4）数据结构说明

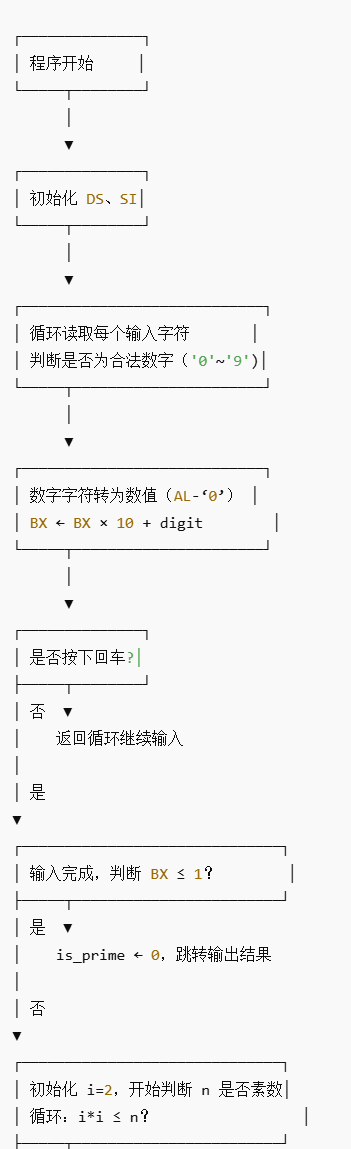
| 名称 | 类型 | 含义与作用 |
| --- | --- | --- |
| buf | 字节数组 | 用于保存用户输入的 10 个一位数字（0–9） |
| max | 字节变量 | 记录当前已输入数据中的最大值 |
| min | 字节变量 | 记录当前已输入数据中的最小值 |
| string1 | 字符串常量 | 提示语 "The maximum value is $"，用于输出最大值 |
| string2 | 字符串常量 | 提示语 "The minimum value is $"，用于输出最小值 |
| prompt | 字符串常量 | 提示信息 "press any key to continue..." |
| 栈 | 隐式结构 | 在 int 21h 调用或子程序调用时使用 |

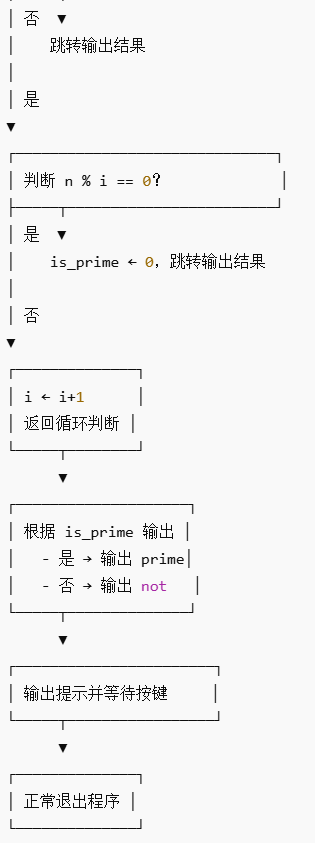
（5）主要符号名

| 符号名 | 类型/作用 | 说明 |
| --- | --- | --- |
| main | 主过程入口 | 相当于 C 语言的 int main() |
| prompt | 数据段字符串 | 显示提示信息 press any key to continue... |
| loop1 | 输入循环入口 | 控制用户输入次数（输入10次） |
| input\_digit\_entrance | 输入合法判断点 | 判断输入字符是否为合法数字字符 '0' ~ '9' |
| less\_than\_min | 分支标签 | 当前输入值小于 min 时跳转执行 |
| greater\_than\_max | 分支标签 | 当前输入值大于 max 时跳转执行 |
| branch\_exit | 分支出口 | 更新完 max 或 min 后的公共出口 |
| loop\_exit | 循环结束标签 | 输入完 10 个数字后跳转至此，准备输出结果 |

* 1. 输入一个不大于65535的十进制非负整数，判断其是否为素数，如果是素数，输出字符串“It’s a prime.”，否则输出字符串“It’s not a prime.”。

（1）.程序流程图：





（2）.C类伪代码：

#include <stdio.h>

unsigned int n, i;

int is\_prime = 1;

int main() {

*// 1. 输入一个十进制正整数*

    n = 0;

    char ch;

    while (1) {

        ch = getchar();

        if (ch == '\r') break;           *// 回车结束*

        if (ch >= '0' && ch <= '9') {

            ch -= '0';

            n = n \* 10 + ch;

        }

    }

*// 2. 判断是否为素数*

    if (n <= 1) {

        is\_prime = 0;

    } else {

        for (i = 2; i \* i <= n; i++) {

            if (n % i == 0) {

                is\_prime = 0;

                break;

            }

        }

    }

*// 3. 输出结果*

    if (is\_prime) {

        printf("It's a prime.\n");

    } else {

        printf("It's not a prime.\n");

    }

    printf("press any key to continue...");

    getchar();

    return 0;

}

（3）程序算法

① 输入处理算法

通过循环读取用户键入的每个数字字符，并减去 '0' 形成数值

使用乘以 10 加上当前位的方法累积构造整数，存入 BX 中

回车ASCII 13表示输入结束

② 素数判断算法

对于 n ≤ 1 的情况直接返回不是素数

枚举 i 从 2 到 √n，判断是否存在 n % i == 0，若存在即非素数

循环中用 MUL 计算 i² 与 DIV 执行除法并比较余数

③ 输出算法

根据素数标志 is\_prime 输出对应字符串

使用 int 21h 的 AH=9 输出整句提示，AH=1 等待按键

程序退出前显示提示字符串，等待任意键

## （4）数据结构说明

| **名称** | **类型** | **含义与作用** |
| --- | --- | --- |
| BX | 16位寄存器 | 用于存放输入的整数值 n |
| SI | 16位寄存器 | 循环变量 i |
| CX | 16位寄存器 | 作为 is\_prime 变量：1 表示是素数，0 表示不是 |
| AL / AH | 寄存器 | AL 用于保存字符输入，AH 用于设置中断号 |
| prime\_msg | 字符串常量 | 输出字符串："It's a prime." |
| not\_prime\_msg | 字符串常量 | 输出字符串："It's not a prime." |
| prompt | 字符串常量 | 输出字符串："press any key to continue..." |

## （5）主要符号名说明

| **符号名** | **类型/作用** | **说明** |
| --- | --- | --- |
| main | 主过程入口 | 程序入口，等价于 C 语言的 int main() |
| input\_loop | 输入循环标签 | 不断读取键盘字符构造多位整数 |
| input\_done | 输入结束标签 | 用户按下回车键后跳转至此开始处理判断 |
| loop\_start | 判断循环开始 | 执行从 i=2 开始的素数判断过程 |
| not\_divisible | 素数候选判断不整除 | 跳转继续循环尝试下一个 i |
| s1 | 输入小于等于 1 处理 | 直接判断非素数 |
| s3 | 判断结束标签 | 跳转输出结果 |
| not\_prime | 输出非素数 | 输出字符串 "It's not a prime." |
| program\_exit | 正常退出标签 | 使用 int 21h / ah=4Ch 正常退出 DOS |

1. 程序代码：

1.计算1+2+3+…+100，并利用21H中断的02H或09号功能调用将十进制结果显示在屏幕上。

;#include <stdio.h>

data segment

    prompt  db "press any key to continue...$"

data ends

code segment

    assume cs:code,ds:data

;int main(){

main proc far

    ;mov ds,data

    mov ax,data

    mov ds,ax

    mov ax,0      ;ax=变量sum

    mov si,1      ;si=变量i

summation\_entrance:

    cmp si,100

    jg  summation\_exit

    add ax,si

    inc si

    jmp summation\_entrance

summation\_exit:

 mov si,0

 ; *//累加和sum整数分离出的各位数字字符,依次入栈*

 ; *//栈中存入的数字字符的个数   si=变量count*

 ;*//从累加和sum中逆序分离出各位数字，并存入数组digits*

 ;do{

split\_digit\_etrance:

 mov bx,10 ;bl=16位除数，ah=变量dight

 cwd    ;convert word to douleworld ax符号位扩展到dx

 idiv bx   ;ax=商，dx=余数    dx：ax=高被除数

 add dl,'0'

 ;push dl ;余数入栈，push,pop只能操作一个字，不能是字节

 push dx  ;dx的值入栈，其中低8位就是分离的数字字符，高8位全为0

 inc si

 ;al=商

 cmp ax,0

 jne  split\_digit\_etrance

 ;*//依次从栈中弹出各个字符*

output\_digit\_entrance:

 dec si

 pop dx    ;栈弹出一个字，低8位是要输出的数字字符，dl=要输出的数字字符

 mov ah,2

 int 21h

 cmp si ,0

 jg  output\_digit\_entrance

 mov dl,13  ;回车符ascll=13

 mov ah,2

 int 21h

 mov dl,10 ;换行符ascll=13

 int 21h

 ;显示字符串“press any key to continue...”

 ;等待输入任意字符

 mov ah,9

 mov dx,offset prompt

 int 21h

 mov ah,1

 int 21h

 mov ah,4ch ;4ch功能号,结束当前程序，返回操作系统dos

 mov al,0   ;al=返回dos的返回值

 int 21h

main endp

code ends

end main

2.利用21H中断的01H号功能调用输入10个一位数字，将其由数字字符转换为相应整数，并依次保存到字节变量BUF。编程求出这10个数中的最大数和最小数，分别存入字节变量MAX和MIN，并分别将其在屏幕上显示出来。

;#include <stdio.h>

data segment

;  char buf[10];

    buf db 10 dup(0)

    max db 0

    min db 0

    string1  db 13,10,"The maximum value is $"

    string2  db 13,10,"The minimum value is $"

    prompt  db "press any key to continue...$"

;char max;

;char min;

data ends

code segment

    assume cs:code,ds:data

main proc far

    mov ax,data

    mov ds,ax ;设置数据段寄存器ds为data段的段地址

;int main(){

 ;*//输入10个一位数，并求极值*

 ;short i=0;

 mov si,0  ;si=变量i

 ;while(i<10){

loop1:

 cmp si,10

 jnl loop\_exit

 ;char ch;  al=变量ch

input\_digit\_entrance:

    ;while((ch=getchar())<'0' || ch>'9');

 mov ah,1

 int 21h

 cmp al,'0'

 jb input\_digit\_entrance

 cmp al,'9'

 ja input\_digit\_entrance

 sub al,'0'

 mov buf[si],al

 cmp si,0

 ;ch-='0';

 ;buf[i]=ch;

 jne less\_than\_min

 ;if(i==0){

 ;  max=ch;

 ;  min=ch;

 ; }

 mov max,al

 mov min,al

 jmp branch\_exit

less\_than\_min:

  ;else if(ch<min){

 cmp al,min

 jnb greater\_than\_max

 mov min,al

 jmp branch\_exit

   ;min=ch;

  ;}

greater\_than\_max:

    cmp al,max

    jna branch\_exit

    mov max,ch

  ;else if(ch>max){

  ; max=ch;

  ;}

 jmp branch\_exit

branch\_exit:

  ;i++;

 inc si

 ;}

 jmp  loop1

loop\_exit:

 ;*//输出极值*

 ;printf("\nThe maximum value is ");

 mov ah,9

 mov dx,offset string1

 int 21h

 mov ah,2

 mov dl,max

 add dl,'0'

 ;putchar(max+'0');

 int 21h

 mov ah,9

 mov dx,offset string2

 int 21h

 mov ah,2

 mov dl,min

 add dl,'0'

 int 21h

 ;printf("\nThe minimum value is ");

 ;putchar(min+'0');

 mov dl,13   ;输出回车符

 int 21h

 mov dl,10   ;输出换行符

 int 21h

 ;putchar('\n');

 ;显示字符串“press any key to continue...”

 ;等待输入任意字符

 mov ah,9

 mov dx,offset prompt

 int 21h

 mov ah,1

 int 21h

 mov ah,4ch ;4ch功能号,结束当前程序，返回操作系统dos

 mov al,0   ;al=返回dos的返回值

 int 21h

 ;return 0;

;}

main endp

code ends

end main

3.输入一个不大于65535的十进制非负整数，判断其是否为素数，如果是素数，输出字符串“It’s a prime.”，否则输出字符串“It’s not a prime.”。

data segment

    prime\_msg      db 13, 10, "It's a prime.$"

    not\_prime\_msg  db 13, 10, "It's not a prime.$"

    prompt         db "press any key to continue...$"

data ends

code segment

    assume cs:code, ds:data

;*#include <stdio.h>*

;int main() {

main proc far

    mov ax, data

    mov ds, ax ; 设置数据段寄存器 ds 为 data 段地址

    ; unsigned int n;

    ; unsigned int i;

    ; int is\_prime = 1;

    mov cx, 1           ; CX = is\_prime 标志变量

    ; 输入十进制整数

    xor bx, bx          ; BX ← 输入结果，初始为 0

input\_loop:

    mov ah, 1

    int 21h             ; 读取字符到 AL

    cmp al, 13          ; 回车？结束输入

    je  input\_done

    cmp al, '0'

    jb  input\_loop

    cmp al, '9'

    ja  input\_loop

    sub al, '0'

    xor ah, ah          ; AL → AX

    push ax

    mov ax,bx

    mov si, 10

    mul si

    pop bx              ; AX = digit \* 10

    add bx, ax          ; BX = BX \* 10 + digit

    jmp input\_loop

input\_done:

    mov si, 2           ; SI = i = 2 每次判断前初始化

    cmp bx, 1

    jbe s1              ; 若 n <= 1，不是素数

    jmp loop\_start

s1:

    mov cx, 0           ; CX = 0 表示不是素数

    jmp s3

; for (i = 2; i\*i <= n; i++)

loop\_start:

    mov ax, si

    mul si              ; AX = i \* i

    cmp ax, bx

    ja  s3              ; i\*i > n → 跳出循环

    mov ax, bx

    xor dx, dx          ; 清除除法高位

    div si              ; AX / SI，余数在 DX

    cmp dx, 0

    jne not\_divisible

    ; n % i == 0 → 非素数

    mov cx, 0

    jmp s3

not\_divisible:

    inc si

    jmp loop\_start

s3:

    cmp cx, 1

    jne not\_prime

    ; 输出：It's a prime.

    mov ah, 9

    mov dx, offset prime\_msg

    int 21h

    jmp program\_exit

not\_prime:

    ; 输出：It's not a prime.

    mov ah, 9

    mov dx, offset not\_prime\_msg

    int 21h

    ; 提示并等待按键

    mov ah, 9

    mov dx, offset prompt

    int 21h

    mov ah, 1

    int 21h

    mov ah, 4ch

    mov al, 0

    int 21h

program\_exit:

    mov ah, 4ch

    int 21h

main endp

code ends

end main

1. 结果分析：

1.计算1+2+3+…+100，并利用21H中断的02H或09号功能调用将十进制结果显示在屏幕上。

（1）.调试情况与问题解决

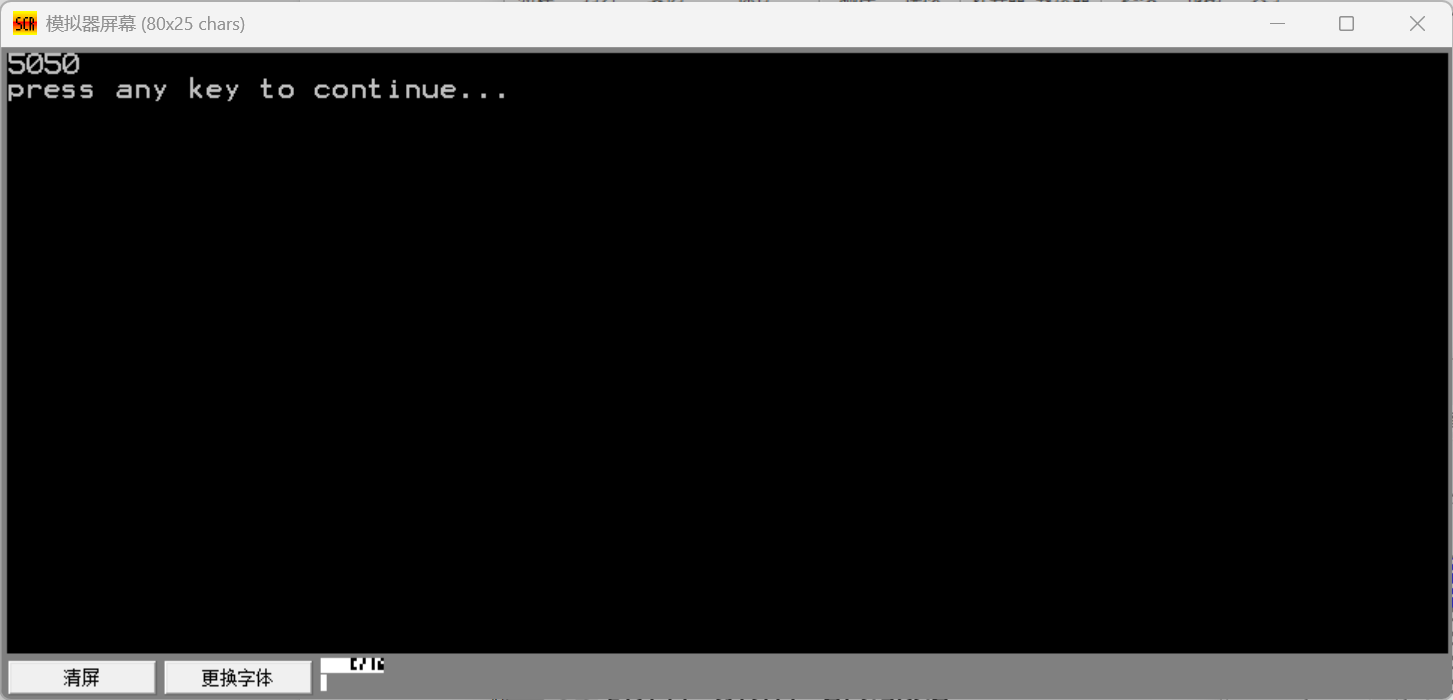
在上机调试过程中，程序总体结构清晰，指令逻辑完整，但也遇到以下典型问题：

①push dl 报错  
问题：8086 架构中 push 和 pop 只能作用于 16 位寄存器，不能直接 push dl。  
解决：使用 push dx，因为 dl 作为字符存在于 dx 的低 8 位中，高 8 位自动为 0，不影响字符输出。

②栈弹出字符后显示乱码  
原因：没有使用 add dl, '0' 将数字转为 ASCII 字符。  
解决：在压栈前对余数加 '0'，确保输出的是字符格式。

③输出顺序错误  
原因：数字字符直接输出会倒序,低位先出，需先全部压入栈后再反向输出。  
解决：正确使用 push dx 入栈、pop dx 出栈实现字符反序显示。

(2).程序输出结果



5050 是从 1 加到 99 的整数和，验证了加法循环逻辑正确。

数字按十进制字符逐位输出，说明除余处理与字符转化正确。

输出字符串和按键等待均正常工作，说明 DOS 中断调用逻辑无误。

(3).现象分析与技巧总结

①栈的利用

本程序利用栈将数字字符逆序存储，再顺序输出，实现了低位先算、高位先显的效果，是处理逆序输出的经典技巧。

②寄存器变量映射

使用 AX 作为求和变量，SI 兼作循环与字符计数器，实现了简洁的寄存器管理。

③分离逻辑清晰

程序结构采用分段标签,如 summation\_entrance、split\_digit\_entrance、output\_digit\_entrance，便于调试和阅读，体现了良好的汇编设计风格。

2.利用21H中断的01H号功能调用输入10个一位数字，将其由数字字符转换为相应整数，并依次保存到字节变量BUF。编程求出这10个数中的最大数和最小数，分别存入字节变量MAX和MIN，并分别将其在屏幕上显示出来。

（1）调试情况与问题解决

在上机调试过程中，程序整体结构较为清晰，功能划分合理，但也遇到了以下几个典型问题：

① 未初始化数据段导致提示字符串乱码或程序异常

问题：若忘记执行 mov ax, data 和 mov ds, ax，数据段变量 prompt、string1、buf 无法正确访问，导致输出乱码或程序异常。

解决：严格按照初始化顺序设置段寄存器 DS，确保数据段访问正常。

② 极值比较逻辑错误

问题：在输入第一个数字时未对 max 和 min 正确初始化，会导致后续极值比较错误或未进入更新分支。

解决：在 i==0 时，显式地将第一个输入值赋给 max 和 min，作为后续比较基准。

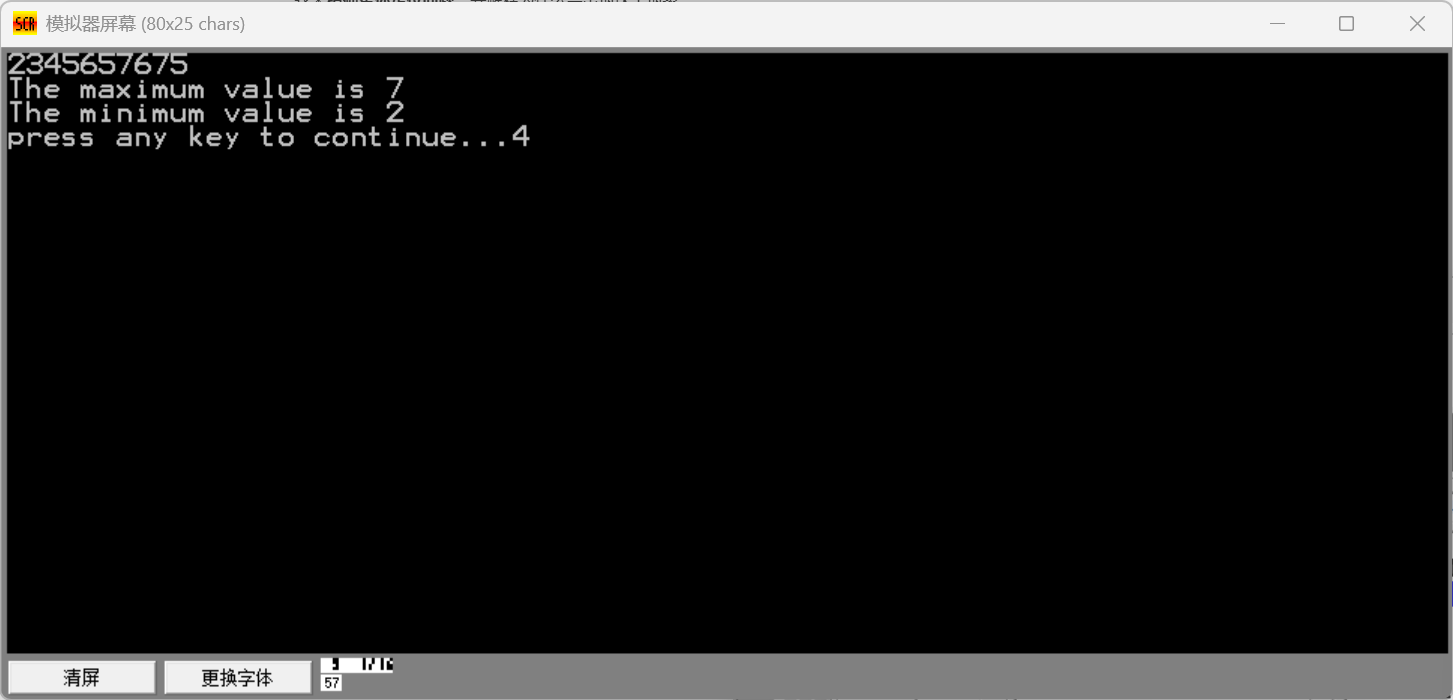
③ 输出字符非 ASCII 格式，结果显示为乱码

问题：将 max 和 min 直接送入 DL 输出，没有加 '0'，导致输出为二进制数值而非字符。

解决：在输出前统一执行 add dl, '0'，将数值转换为对应 ASCII 字符再调用中断输出。

（2）程序输出结果

程序运行时，输入任意合法的 10 个一位十进制数字,范围 '0'~'9'后，控制台输出如下：



输出的最大值与最小值准确反映输入数据中的极值，说明：

buf[] 中存储正确；

极值比较与更新逻辑正确；

数字到字符的转换无误；

字符串输出、等待按键功能正常，验证了 int 21h DOS 中断服务调用正确。

（3）现象分析与程序技巧总结

① 栈操作简化字符反序问题  
虽然本程序没有对数字分离入栈，但通过 buf[si] 顺序存储输入、逐次比较极值，逻辑上等价于前面实验中的“反序输出”方法。在输出提示信息前适当使用 CR（13）与 LF（10）也体现了 DOS 下的换行标准。

② 寄存器变量映射清晰高效

使用 SI 作为循环变量 i；

使用 AL 暂存字符输入，DL 输出字符，AH 控制 DOS 功能；

MAX 与 MIN 用作全局变量，存储极值；  
这些都充分体现了汇编中寄存器与数据段变量合理配合的技巧。

③ 程序结构清晰，模块划分良好  
采用了多个明确的标签分段控制程序流程，包括：

| 标签名 | 功能 |
| --- | --- |
| loop1 | 输入主循环 |
| input\_digit\_entrance | 判断输入合法性 |
| less\_than\_min | 当前值小于最小值时的更新分支 |
| greater\_than\_max | 当前值大于最大值时的更新分支 |
| branch\_exit | 比较后跳转统一出口 |
| loop\_exit | 输入完成后跳转到结果输出部分 |

这种分支结构便于调试和维护，具有良好的程序可读性。

3.输入一个不大于65535的十进制非负整数，判断其是否为素数，如果是素数，输出字符串“It’s a prime.”，否则输出字符串“It’s not a prime.”。

（1）调试情况与问题解决

在本次实验的上机调试过程中，程序总体结构合理，功能划分明确，但在初期实现和多次测试过程中也暴露出以下几个典型问题：

① 输入字符转换逻辑错误

问题描述：初始版本中使用 mul si 进行输入构造时，错误地将数字字符乘以 10，而非当前累积数，导致 BX 中结果始终为 0 或异常值。

解决方法：将输入逻辑修改为 BX = BX \* 10 + digit，并正确使用 mul bx 而非 mul si，确保每位输入按权累加。

② is\_prime 标志变量被破坏

问题描述：最初使用 CX 作为乘法常数 10 保存位置，之后又将其作为布尔变量 is\_prime 使用，导致逻辑判断错乱，所有数都被判断为非素数。

解决方法：重构乘法部分，使用 SI 存放常数 10，避免改动 CX，从而保证布尔标志值在程序判断中不被意外覆盖。

③ 除法前未清零 DX 导致余数错误

问题描述：在素数判断过程中使用 DIV 指令前未执行 xor dx, dx，导致 DX 残留旧值参与除法，结果余数 n % i 错误，导致误判。

解决方法：每次 mov ax, bx 后立即执行 xor dx, dx，保证除法前高位清零，确保除法逻辑结果准确。

④ 输入循环未重置 i 值导致跳过判断

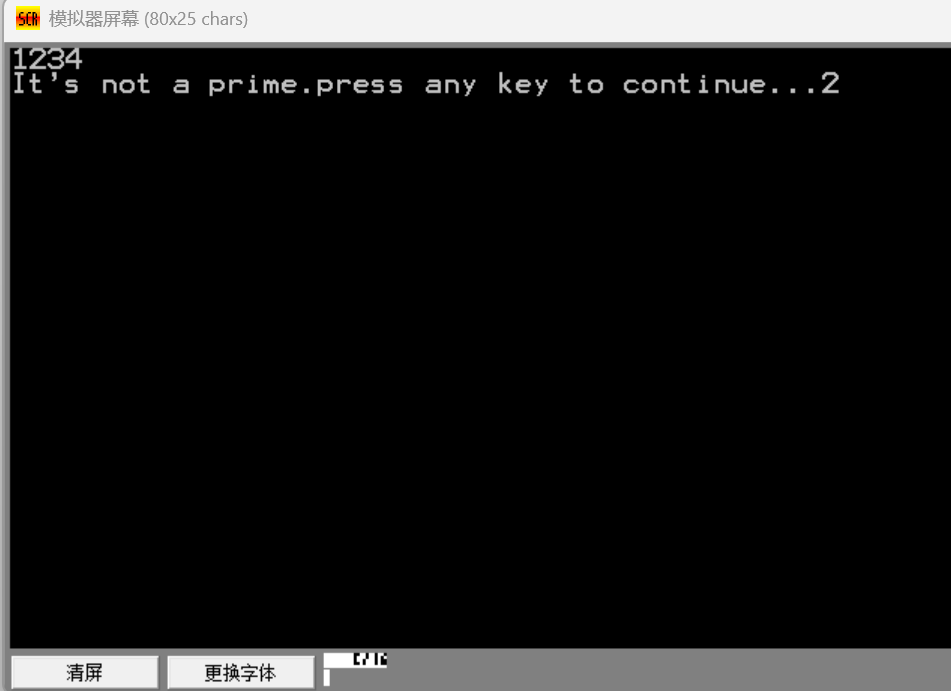
问题描述：若连续多次运行程序逻辑，循环变量 SI 在上一次判断后未重置，导致 i\*i > n 初始即成立，跳过整个判断逻辑。

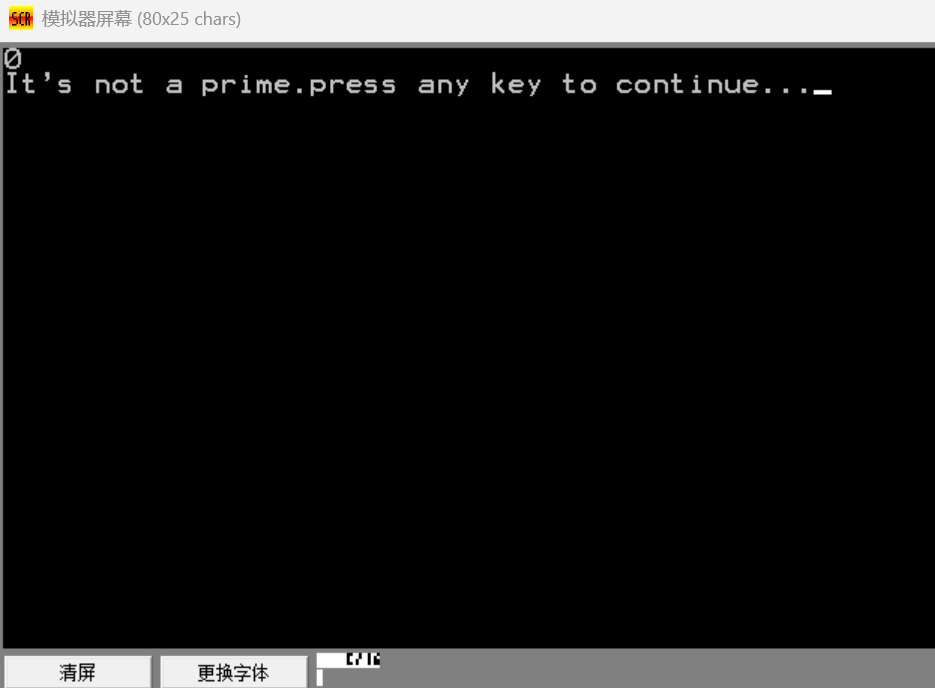
解决方法：在 input\_done: 标签处重新初始化 SI ← 2，确保每次判断都从正确的起点开始。

（2）程序输出结果

程序运行后，从键盘输入任意合法十进制整数，如：9973等，程序将自动判断是否为素数并输出结果。实际运行中，以下输入输出情况说明程序逻辑准确、功能完整：









程序输出信息准确，素数判断结果与理论一致，验证了以下功能正确实现：

多位十进制数值输入与累加

i\*i ≤ n 的循环逻辑正确

% 取模结果判断准确

（3）现象分析与程序技巧总结

① 输入十进制转数值的乘加构造法

使用 BX = BX \* 10 + digit 实现多位十进制转整型，是将 getchar 逐位输入映射为数值的经典写法，避免了使用栈进行数字反序，便于在汇编中实现。

② CX / SI / BX 等寄存器职责明确

CX 被明确用于 is\_prime 标志变量，判断过程只读不写；

SI 同时作为乘法常数和循环变量；

BX 保存最终输入值 n；

AX、DX 辅助参与乘除法；  
这种寄存器映射方式清晰明了，是汇编中变量寄存器选择与复用技巧的体现。

③ 使用 DIV 判断素性具代表性

通过 DIV 和余数 DX 判断是否能整除，体现出汇编语言在低层次精确控制运算过程的优势；

每次判断都需清空 DX，也体现了对寄存器状态管理的严谨性。、

④ DOS 中断调用结构清晰、功能合理

AH=1 读取字符，AH=9 输出字符串，AH=4Ch 正常退出；

输出结构为“主语句 + 结果字符 + 换行提示”，符合 DOS 程序交互习惯；

最后提示 press any key to continue... 具有良好人机交互体验。

⑤ 程序流程标签划分清晰

| 标签名 | 功能说明 |
| --- | --- |
| input\_loop | 逐位读取十进制输入字符 |
| input\_done | 用户按回车结束输入，准备判断 |
| loop\_start | 素数判断过程 i\*i ≤ n 的主循环入口 |
| not\_divisible | n % i ≠ 0 时跳转继续下一轮判断 |
| s1 | 输入值 ≤ 1 的特判情况处理 |
| s3 | 判断逻辑结束，准备根据 is\_prime 输出 |
| not\_prime | 输出非素数提示并等待按键 |
| program\_exit | 程序正常退出 |

这种流程划分方式利于调试、阅读和模块化维护，是优秀汇编程序结构设计的体现。

4.实验心得体会

（1）本次实验让我切身体验到汇编语言与高级语言的区别。每一条指令都需要精确控制，必须手动处理数据段初始化、寄存器使用、跳转逻辑等细节，增强了我对程序运行底层机制的认识。

（2）通过对每位数字字符的输入、减 '0' 并乘 10 累加，构建了 BX 中的数值。这一过程中不仅理解了字符与数值的本质差异，也学会了在汇编中如何实现进位加法的逐位输入。

（3）使用 INT 21H 的多个功能子号，AH=1, 2, 9, 4Ch，实现了字符输入、输出、字符串显示与程序退出。让我对 DOS 下汇编程序与操作系统之间的接口机制有了更直观的理解。

（4）在程序中合理使用了 AX、BX、CX、SI、DL 等寄存器，分别承担输入、累加、判断标志、循环变量与输出的作用。我体会到寄存器分工明确能显著提升程序稳定性与可读性。

（5）程序中运用了 CMP、JBE、JA、JNE、JE 等条件跳转指令，配合标签实现 if-else 分支与 for 循环功能。通过本次实践，我深入理解了汇编控制流的构建方式。

（6）实验过程中出现了如除法前未清除 DX、误用乘法寄存器、标志变量被覆盖等问题。通过逐步排查与单步跟踪，我掌握了如何定位逻辑错误并修复程序，是一次非常锻炼思维的过程。

（7）将判断一个整数是否为素数的算法转换为指令级的实现，让我更清楚地理解了算法—流程—指令三者之间的关系，提升了将抽象问题形式化、结构化的能力。