**实验二 分支与循环程序设计**

班级： 22计科3班 姓名：张雅瑞 学号： 2022334323029

1. **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**

本次实验你所完成的具体内容加以说明。

程序一

这个汇编实验的主要目标是通过循环计算累加和，并将结果以十进制字符的形式输出在屏幕上。下面简要总结这个实现的主要思路和关键步骤：

**初始化**

1. **数据段设置**：定义用于累加的变量和用于显示的信息字符串。
2. **代码段设置**：设置程序入口，初始化数据段，并输出提示信息。

**循环累加**

1. **设置循环次数**：使用循环结构，将当前循环计数 **cx** 加到累加和 **sum** 中。
2. **执行循环**：通过 **loop** 指令控制循环，逐次累加，直到循环次数归零。

**将累加和转换为十进制**

1. **准备转换**：将 **sum** 存储到 **ax**，然后通过除法来将其转换为十进制。
2. **转换为十进制**：通过除数为10的除法，将商和余数分离，余数即为十进制位。将余数存储到堆栈，并继续除以10，直到商为零。

**显示累加和结果**

1. **输出字符串**：输出提示信息，表示接下来将显示累加和。
2. **输出十进制字符**：从堆栈中弹出余数，转换为ASCII字符并输出，直到所有数字输出完毕。

**结束程序**

1. **结束流程**：调用DOS中断，结束程序。

**主要思路总结**

整个实验的实现思路围绕着循环和累加展开：

* 使用循环结构来累计指定次数的数值。
* 利用除以10的方式，将累加和转换为十进制字符，并通过堆栈来保存和输出每个十进制位。
* 利用DOS中断实现字符串和字符的输出，以及程序的结束。

这个思路实现了一个基本的循环和累加操作，并展示了在8086汇编中如何使用中断和堆栈操作实现十进制输出。通过这种方式，你可以将累加和结果以可读的形式显示在屏幕上。

程序二

在实验二中，通过汇编代码实现了读取10个一位数字，将其转换为整数，并将其依次存储在 **buf** 变量中。程序会计算这10个数字中的最大数和最小数，并分别显示它们在屏幕上。以下是对该实验的内容和实现过程的简要说明。

**代码逻辑概述**

1. **数据段初始化**：定义 **buf** 用于存储10个输入的数字；**max** 和 **min** 用于存储最大值和最小值；定义提示信息字符串，用于显示输入提示和结果。
2. **堆栈段初始化**：分配10个字节的空间，作为堆栈。
3. **代码段设置**：初始化数据段和堆栈段，设置程序入口点。
4. **输入10个字符**：使用DOS中断21H的01号功能，读取10个一位数字，将其存储在 **buf** 中。
5. **计算最大值和最小值**：通过循环比较 **buf** 中的值，找到最大值和最小值，并存储在 **max** 和 **min** 中。
6. **显示最大值和最小值**：使用DOS中断21H的09号和02号功能，将最大值和最小值显示在屏幕上。

**实现过程说明**

1. **数据段和堆栈段初始化**：
   * **buf** 用于存储10个一位数字。
   * **max** 和 **min** 用于存储最大值和最小值。
   * **Str1** 用于提示用户输入数字。
   * **Str2** 和 **Str3** 用于显示最大值和最小值提示。
2. **读取10个字符**：
   * 使用循环，调用DOS中断21H的01号功能，读取用户输入的10个一位数字，并将其存储在 **buf** 中。
3. **计算最大值和最小值**：
   * 使用循环，比较 **buf** 中的值，找到最大值和最小值。
   * 初始时，**al** 存储第一个数字，**bl** 存储最小值。
   * 在循环中，通过比较和条件跳转更新最大值和最小值。
4. **显示最大值和最小值**：
   * 使用DOS中断21H的09号功能，显示提示信息。
   * 使用DOS中断21H的02号功能，显示单个字符。
   * 从 **max** 和 **min** 中获取值，并将其转换为ASCII字符后输出。

**结束程序**

1. **结束程序**：使用 **ret** 指令返回，并使用DOS中断21H的4C号功能结束程序。

**实验总结**

在这个实验中，通过读取10个一位数字，将其转换为整数，并存储在 **buf** 中。程序通过循环比较计算了最大值和最小值，然后分别显示在屏幕上。代码使用了循环、DOS中断、条件跳转、堆栈操作等关键操作，展示了汇编语言中数据处理和输出的基础技能。

程序三

实验三的目标是让用户输入一个不大于 65535 的十进制非负整数，然后判断这个数是否是素数。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.”。以下是对这个实验的具体内容和实现过程的简要说明。

**程序的主要逻辑**

1. **数据段和堆栈段**：定义数据段来存储输入的数值、提示信息和结果输出字符串。堆栈段用于存储堆栈数据。
2. **初始化**：设置数据段和堆栈段，并输出提示信息，要求用户输入一个不大于 65535 的十进制非负整数。
3. **读取输入**：通过循环结构和 DOS 中断 21H 的 01 号功能，读取用户输入的十进制数字，并转换为整数。
4. **判断素数**：通过循环判断给定数是否是素数。如果是素数，输出提示信息 “It’s a prime.”；否则，输出 “It’s not a prime.”。
5. **输出结果**：使用 DOS 中断 21H 的 09 号功能输出字符串，显示判断结果。

**实现过程说明**

1. **初始化数据段和堆栈段**：
   * 使用 **MOV AX, DATA** 和 **MOV DS, AX** 初始化数据段。
   * 使用 **MOV AX, STACK** 和 **MOV SS, AX** 初始化堆栈段。
   * 输出提示信息 “Please input a number:”。
2. **读取用户输入**：
   * 使用循环读取用户输入的数字字符，并将其转换为整数。
   * 利用 **SUB AL, 30H** 将 ASCII 数字字符转换为整数，然后将其存储在 **NUMBER** 中。
3. **判断素数**：
   * 使用循环判断给定数是否是素数。
   * 从 2 开始，检查给定数是否可以被任何数整除。
   * 如果找不到除数，使得给定数整除，则这个数是素数；否则，这个数是非素数。
4. **输出结果**：
   * 如果是素数，输出字符串 “It’s a prime.”。
   * 如果不是素数，输出字符串 “It’s not a prime.”。

**程序结构及符号说明**

* **数据段**：定义存储输入的变量 **NUMBER**，存储提示信息的字符串 **STRING0**，以及输出结果的字符串 **STRING1** 和 **STRING2**。
* **堆栈段**：定义堆栈空间。
* **代码段**：定义程序的主要流程和逻辑，包括输入、判断、输出结果等。

**代码模块及流程**

* **输入**：循环读取用户输入的十进制数字，并存储在 **NUMBER** 中。
* **判断素数**：通过循环和除法判断给定数是否是素数。
* **输出结果**：使用 DOS 中断输出字符串，显示判断结果。
* **结束程序**：调用 DOS 中断结束程序。

**实验成果**

这个实验的实现过程中，完成了如下工作：

* 通过 DOS 中断读取用户输入。
* 使用循环结构判断给定数是否是素数。
* 使用 DOS 中断输出字符串，显示判断结果。
* 使用条件跳转和循环结构，判断并输出结果。

通过这个实验，成功实现了对十进制非负整数的素数判断，并使用汇编语言完成了读取输入、判断逻辑和输出结果的过程。

**三、设计思想**

绘制程序框图，并说明原理及算法、程序及数据结构、主要符号名等。

程序一

**程序框图**

程序框图描述了代码的主要流程，包括初始化、循环计算累加和、十进制转换、输出结果以及结束程序的步骤。

|  |
| --- |
| +---------------------+  | 程序开始 |  +---------------------+  |  v  +---------------------+  | 数据段初始化 |  +---------------------+  |  v  +---------------------+  | 循环计算累加和 |  +---------------------+  |  v  +---------------------+  | 转换为十进制 |  +---------------------+  |  v  +---------------------+  | 显示提示信息 |  +---------------------+  |  v  +---------------------+  | 输出结果的十进制字符 |  +---------------------+  |  v  +---------------------+  | 结束程序 |  +---------------------+ |

程序开始->数据段初始化 -> 循环计算累加和->转换为十进制->显示提示信息->输出结果的十进制字符->结束程序

**原理及算法**

实验的主要原理和算法包括：

* **循环计算**：通过循环结构，将当前循环计数器 **cx** 的值加到累加和 **sum** 中，实现简单的累加操作。
* **十进制转换**：通过除以10的方式，将累加和 **sum** 转换为十进制。利用堆栈存储余数，构成十进制数字。
* **输出字符串**：利用DOS中断 21H 的功能号来输出字符串和字符，将计算结果显示在屏幕上。
* **结束程序**：使用DOS中断结束程序。

**程序及数据结构**

程序结构包括数据段和代码段，其中数据段包含用于存储累加和的变量，以及提示信息等字符串。代码段包含程序的主要逻辑和流程。

* **数据段**：定义了 **sum**、**message** 等变量和字符串。
* **代码段**：包含程序的入口 **start**、循环计算累加和的逻辑 **calculate\_sum**、十进制转换的逻辑 **convert\_to\_ascii**、输出结果的逻辑 **display\_result** 以及结束程序的步骤。

**主要符号名**

* **变量**：
  + **sum**：用于存储累加和的变量。
* **字符串**：
  + **message**：输出的提示信息字符串。
* **寄存器**：
  + **cx**：循环计数器。
  + **ax**、**bx**、**dx**：用于运算和数据存储。
* **代码段标签**：
  + **calculate\_sum**：用于循环累加的标签。
  + **convert\_to\_ascii**：用于十进制转换的标签。
  + **display\_result**：用于输出结果的标签。

**总结**

实验通过简单的循环计算累加和，并将结果转换为十进制输出在屏幕上。整个过程包括初始化、循环计算、十进制转换、字符串输出、结果显示和程序结束。代码中利用循环、堆栈操作和DOS中断来实现这一过程。

程序二

实验二的目标是读取10个一位数字，将其转换为整数，并将其存储在字节变量 **buf** 中，然后找到这10个数中的最大值和最小值，分别存储在 **max** 和 **min** 中，并将其在屏幕上显示。以下是实验的程序框图、原理及算法、程序及数据结构、主要符号名等的详细说明。

**程序框图**

框图描述了实验二的主要流程，从初始化到输出结果，再到结束程序。

|  |
| --- |
| +---------------------------------+  | 初始化段寄存器 |  +---------------------------------+  |  v  +---------------------------------+  | 初始化数据段和堆栈段 |  +---------------------------------+  |  v  +---------------------------------+  | 读取10个字符并存储到 `buf` |  +---------------------------------+  |  v  +---------------------------------+  | 计算最大值和最小值 |  +---------------------------------+  |  v  +---------------------------------+  | 显示最大值和最小值 |  +---------------------------------+  |  v  +---------------------------------+  | 结束程序 |  +---------------------------------+ |

初始化段寄存器->初始化数据段和堆栈段->读取10个字符并存储到 `buf`->计算最大值和最小值->显示最大值和最小值->结束程序

**原理及算法**

实验二的原理是通过DOS中断读取输入，并通过循环找到最大值和最小值。算法的主要部分包括：

* **循环读取输入**：通过循环结构，读取10个字符，将其存储在 **buf** 中。
* **循环比较找到最大值和最小值**：通过循环比较 **buf** 中的字符，找到最大值和最小值。
* **输出结果**：使用DOS中断将提示信息和结果显示在屏幕上。
* **结束程序**：通过DOS中断结束程序。

**程序及数据结构**

实验的主要数据结构和程序流程如下：

* **数据段**：包含用于存储输入的 **buf**，以及存储最大值和最小值的 **max** 和 **min**。还包含用于显示提示信息的字符串 **Str1**、**Str2** 和 **Str3**。
* **堆栈段**：定义了用于堆栈的空间。
* **代码段**：包含程序的主要逻辑和流程。

**主要符号名**

* **变量**：
  + **buf**：用于存储10个输入字符的字节数组。
  + **max**：用于存储最大值的变量。
  + **min**：用于存储最小值的变量。
* **字符串**：
  + **Str1**：用于提示用户输入的字符串。
  + **Str2**：用于显示最大值的字符串。
  + **Str3**：用于显示最小值的字符串。
* **寄存器**：
  + **cx**：循环计数器，用于控制循环。
  + **si**：源索引寄存器，用于指向 **buf** 中的元素。
  + **ax**, **bx**, **dx**：用于存储和计算数据。
* **代码段标签**：
  + **lop1**：用于循环读取输入的标签。
  + **lop2**：用于循环计算最大值和最小值的标签。

**代码模块及流程**

* **初始化**：设置数据段和堆栈段，准备读取输入。
* **读取输入**：使用DOS中断21H的01号功能，读取10个字符，并存储在 **buf** 中。
* **计算最大值和最小值**：通过循环比较，找到最大值和最小值，分别存储在 **max** 和 **min** 中。
* **输出结果**：通过DOS中断输出提示信息和计算结果。
* **结束程序**：使用DOS中断21H的4C号功能结束程序。

**总结**

实验二的实现通过循环结构和DOS中断实现了输入、计算和输出的流程。代码中使用了循环、堆栈操作、条件跳转等关键操作。通过这个实验，你可以了解如何在汇编中实现数据输入、计算和输出的完整流程。

程序三

**程序框图**

以下是实验三的程序框图，展示了代码的主要流程和逻辑结构。

|  |
| --- |
| +---------------------------------+  | 初始化段寄存器 |  +---------------------------------+  |  v  +---------------------------------+  | 提示用户输入 |  +---------------------------------+  |  v  +---------------------------------+  | 循环读取10个一位数字，转换为整数 |  +---------------------------------+  |  v  +---------------------------------+  | 判断是否为素数 |  +---------------------------------+  |  v  +---------------------------------+  | 输出结果 |  +---------------------------------+  |  v  +---------------------------------+  | 结束程序 |  +---------------------------------+ |

**原理及算法**

实验三的主要原理和算法包括：

* **循环输入**：通过循环结构读取10个字符，并转换为整数。
* **判断素数**：通过循环和除法，检查给定数是否是素数。
* **输出结果**：使用 DOS 中断输出结果字符串，告知用户该数是否为素数。

**程序及数据结构**

实验三的主要程序结构和数据结构如下：

* **数据段**：定义变量 **NUMBER** 用于存储输入的十进制数；**STRING0**、**STRING1**、**STRING2**、**STRING3** 用于存储提示信息和结果信息。
* **堆栈段**：分配一定空间作为堆栈。
* **代码段**：包含主要的程序流程和逻辑，包括输入、判断和输出。

**主要符号名**

* **变量**：
  + **NUMBER**：用于存储输入的十进制整数。
  + **CNT**：计数器，用于控制循环和判断素数。
* **字符串**：
  + **STRING0**：提示用户输入的字符串。
  + **STRING1**：输出素数提示的字符串。
  + **STRING2**：输出非素数提示的字符串。
  + **STRING3**：非法字符提示字符串。
* **寄存器**：
  + **cx**：循环计数器。
  + **ax**, **bx**, **dx**：用于数据存储和计算。
  + **si**：用于数组索引。
* **代码段标签**：
  + **INPUT**：用于循环输入的标签。
  + **OUTSIDE**：用于处理结束输入的标签。
  + **NEXT**：用于判断素数的标签。
  + **ISPRIME**：素数判断成功的标签。
  + **NOTPRIME**：素数判断失败的标签。
  + **TAIL**：程序结束部分。

**代码模块及流程**

* **初始化**：设置数据段和堆栈段，输出提示信息，要求用户输入十进制数。
* **读取输入**：通过循环读取10个字符，将其转换为整数。
* **判断素数**：通过循环和除法，检查给定数是否是素数。
* **输出结果**：使用 DOS 中断输出结果字符串，显示是否是素数。
* **结束程序**：调用 DOS 中断结束程序。

**程序结构和逻辑**

这个实验的结构包括数据段、堆栈段和代码段。主要逻辑包括以下步骤：

1. **输入**：通过循环读取10个一位数字，并转换为整数。
2. **判断**：使用循环和条件判断，确定是否是素数。
3. **输出**：根据判断结果输出提示信息，告知用户该数是否是素数。
4. **结束程序**：调用 DOS 中断结束程序。

**实验总结**

实验三通过读取用户输入，判断给定十进制数是否是素数，并输出结果。代码中利用循环、除法、条件跳转、DOS 中断等关键操作，实现了完整的流程。通过这个实验，可以了解汇编语言中如何进行数据输入、判断逻辑和输出结果的实现。

**四、程序代码**

程序清单。

程序一

|  |
| --- |
| .model small  .stack 100h    .data  sum DW 0 ; 用于存储累加和的变量  message DB 'The sum is: $' ; 显示的消息    .code  start:  mov ax, @data  mov ds, ax    mov cx, 10 ; 设置循环次数为10  ;xor sum, sum ; 初始化累加和为0  mov sum,0    calculate\_sum:  add sum, cx ; 将cx（当前循环次数）加到sum上  loop calculate\_sum ; 如果cx不为0，则减少cx并跳转回calculate\_sum    ; 将结果转换为ASCII并显示  mov ax, sum ; 将和加载到ax中  cwd ; 扩展为32位，以便处理可能的负数  mov bx, 10 ; 除数，用于转换为十进制  xor cx, cx ; 清零cx，用作计数器    convert\_to\_ascii:  xor dx, dx ; 清零dx，准备除法  div bx ; 除以10，商在ax中，余数在dl中  push dx ; 保存余数（个位数）  inc cx ; 增加计数器  test ax, ax ; 检查商是否为0  jnz convert\_to\_ascii ; 如果不为0，继续转换    ; 显示消息前缀  mov ah, 9  mov dx, offset message  int 21h  ;显示数字  display\_result:  pop dx ; 弹出并显示个位数  add dl, '0' ; 转换为ASCII码  mov ah, 2 ; DOS功能号，显示字符  int 21h  loop display\_result ; 如果还有更多的数字要显示，继续循环    ; 结束程序  mov ah, 4ch  int 21h    end start |

程序二

|  |
| --- |
| data segment ; 数据段定义开始  buf db 10 dup(0) ; 存储输入的10个字符  max db 0 ; 最大值  min db 0 ; 最小值  Str1 db 0Dh, 0Ah, 'Please input: $' ; 提示输入  Str2 db 0Dh, 0Ah, 'MAX: $' ; 最大值提示  Str3 db 0Dh, 0Ah, 'MIN: $' ; 最小值提示  data ends ; 数据段定义结束  stack segment stack ; 堆栈段定义  db 10 dup(0) ; 分配10字节堆栈  stack ends ; 堆栈段结束  code segment ; 代码段定义  main proc far  assume cs:code, ds:data, ss:stack  start: ; 初始化段寄存器  push ds ; 保存数据段寄存器  sub ax, ax ; 清零ax  push ax ; 保存ax到堆栈  mov ax, data ; 数据段地址  mov ds, ax ; 设置ds  mov ax, stack ; 堆栈段地址  mov ss, ax ; 设置ss  mov cx, 10 ; 设置循环次数  mov si, 0 ; 初始化si用于存储输入  mov bx, 0 ; 初始化bx  ; 读取输入  lop1:  lea dx, Str1 ; 加载提示信息  mov ah, 09h ; DOS字符串输出功能  int 21h ; 执行中断，显示提示  mov ah, 01h ; DOS字符输入  int 21h ; 读取字符  mov buf[si], al ; 存储输入字符  inc si ; si自增  loop lop1 ; 循环10次  ; 计算最大值和最小值  mov cx, 9 ; 重置循环次数  mov si, 0 ; 重置si  mov al, buf[si] ; 读取第一个字符  mov bl, al ; 初始化bl用于最小值  lop2:  inc si ; 移动到下一个字符  mov dl, buf[si] ; 读取当前字符  cmp dl, al ; 检查最大值  jl update\_max ; 如果dl < al，更新最大值  mov al, dl ; 复制dl到al  update\_max:  cmp dl, bl ; 检查最小值  jge continue\_loop ; 如果dl >= bl，继续  mov bl, dl ; 更新最小值  continue\_loop:  loop lop2 ; 循环结束后，停止  ; 将最大值和最小值存储  mov max, al ; 更新最大值  mov min, bl ; 更新最小值  ; 输出最大值  lea dx, Str2 ; 最大值提示  mov ah, 09h ; DOS字符串输出  int 21h ; 执行中断  mov dl, max ; 显示最大值  mov ah, 02h ; DOS字符输出  int 21h ; 输出字符  ; 输出最小值  lea dx, Str3 ; 最小值提示  mov ah, 09h ; DOS字符串输出  int 21h ; 执行中断  mov dl, min ; 显示最小值  mov ah, 02h ; DOS字符输出  int 21h ; 输出字符  ; 结束程序  ret ; 返回调用点  main endp ; main过程结束  code ends ; 代码段结束  end start ; 指定程序入口点 |

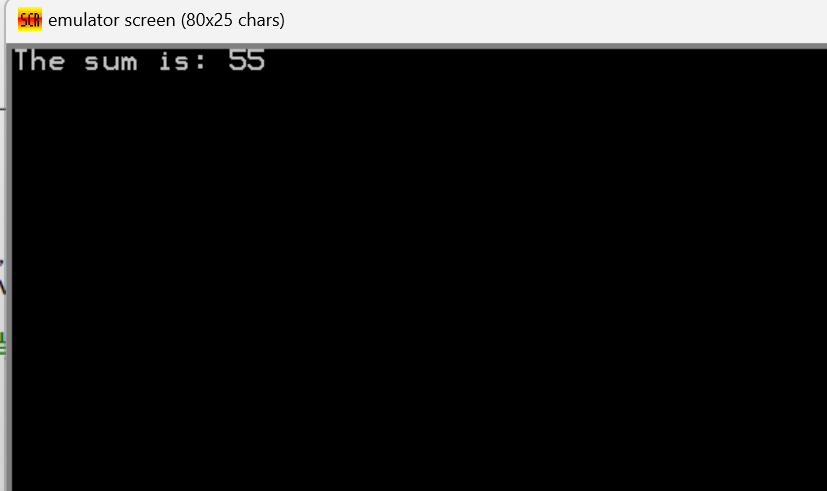
程序三

|  |
| --- |
| ASSUME CS:CODE,DS:DATA ; 假设CS为代码段，DS为数据段    DATA SEGMENT  NUMBER DW 0  STRING0 DB 'PLEASE INPUT A NUMBER: $' ; 定义一个字符串提示输入一个数字  STRING1 DB 'IT IS PRIME $' ; 定义一个字符串表示输入的数字是奇数  STRING2 DB 'IT IS NOT PRIME $' ; 定义一个字符串表示输入的数字是偶数  STRING3 DB 'INLEGAL CHARACTER!$' ; 定义一个字符串表示输入了非法字符  CNT DW 1;  DATA ENDS    CODE SEGMENT  START:  MOV AX,DATA ; 将数据段地址赋值给AX  MOV DS,AX ; 将AX的内容赋值给DS，即初始化数据段  LEA DX,STRING0 ; 加载STRING0的地址到DX，准备输出该字符串  MOV AH,09H ; DOS 21H中断的9号功能，字符串输出  INT 21H ; 调用DOS中断    MOV CX,10 ; 初始化CX为10，用于后续循环读取字符  INPUT:  MOV AH,01H ; DOS 21H中断的1号功能，从键盘读取一个字符  INT 21H ; 调用DOS中断  CMP AL,039H ; 比较输入的字符是否大于'9'（ASCII码）  JA INLEGAL ; 如果是，跳转到非法字符处理  CMP AL,0DH ; 比较输入的字符是否为回车（ASCII码）  JE OUTSIDE ; 如果是，跳转到处理回车    SUB AL, 30H ; 将ASCII字符转为整数  XCHG AX, BX ; 交换AX和BX的内容  MOV DX, 10 ; 将10加载到DX  MUL DX ; 将AX乘以10  XCHG AX, BX ; 交换AX和BX的内容  MOV AH, 0 ; 将AH清零  ADD BX, AX ; 将AX的值加到BX，更新BX为输入的数字  LOOP INPUT ; 如果CX不为0，继续循环输入    INLEGAL:  CALL CRLF ; 调用CRLF过程，输出回车换行  LEA DX,STRING3 ; 加载STRING3的地址到DX，准备输出非法字符提示  MOV AH,09H ; DOS 21H中断的9号功能，字符串输出  INT 21H ; 调用DOS中断  JMP TAIL ; 跳转到程序尾部  OUTSIDE:  CALL CRLF ; 调用CRLF过程，输出回车换行  MOV NUMBER, BX ; 将输入的数放进NUMBER    NEXT:  INC CNT ; 每次循环就加1  MOV AX, CNT ; 将CNT放进AX中  CMP AX, NUMBER ; 如果NUMBER为2，直接跳转到ISPRIME  JE ISPRIME      MOV AX, CNT  MUL CNT ; 这里计算CNT的平方    CMP NUMBER, AX ; 判断是否已经遍历所有可能  JL ISPRIME ; 如果已经遍历的所有可能则跳转到ISPRIME    MOV AX, NUMBER ; 否则将NUMBER放进AX，然后除以CNT，判断余数是否为0  DIV CNT  CMP DX, 0 ; 判断余数是否为0, 为0说明不是素数  JNZ NEXT ; 不为0继续循环  NOTPRIME:  LEA DX,STRING2 ; 加载STRING2的地址到DX，准备输出偶数提示  MOV AH,09H ; DOS 21H中断的9号功能，字符串输出  INT 21H ; 调用DOS中断  JMP TAIL ; 跳转到程序尾部    ISPRIME:  LEA DX,STRING1 ; 加载STRING1的地址到DX，准备输出奇数提示  MOV AH,09H ; DOS 21H中断的9号功能，字符串输出  INT 21H ; 调用DOS中断  JMP TAIL ; 跳转到程序尾部      TAIL:  MOV AX,4C00H ; 准备结束程序，将AX设置为4C00H  INT 21H ; 调用DOS中断结束程序    CRLF PROC NEAR  PUSH AX ; 保存AX的内容到堆栈  PUSH DX ; 保存DX的内容到堆栈  MOV DL,0AH ; 准备输出换行符  MOV AH,2H ; DOS 21H中断的2号功能，字符输出  INT 21H ; 调用DOS中断  MOV DL,0DH ; 准备输出回车符  MOV AH,2H ; DOS 21H中断的2号功能，字符输出  INT 21H ; 调用DOS中断  POP DX ; 从堆栈恢复DX的内容  POP AX ; 从堆栈恢复AX的内容  RET ; 返回调用点  CRLF ENDP    CODE ENDS  END START ; 程序结束，指定入口点为START |

**五、结果分析**

**1.程序的输出结果及对结果的分析。**

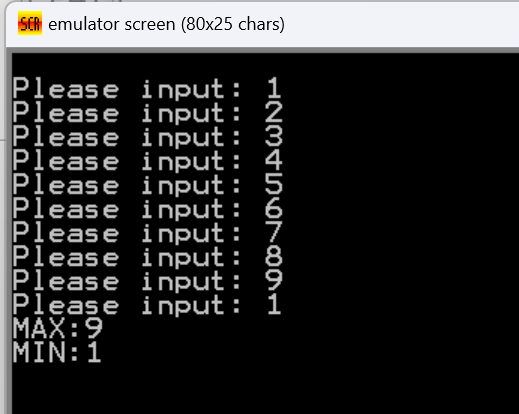
程序一



根据输出结果，可以对代码的各个部分进行分析，以确保其逻辑和运作正常：

1. **循环累加**：通过 **add sum, cx** 实现累加，每次循环后 **cx** 减1，直到 **cx** 为零。通过 **loop** 指令实现了这一过程。
2. **十进制转换**：通过将累加和 **sum** 转换为十进制，确保正确显示结果。这个过程使用除以10的方法，将结果转换为个位、十位等，并通过堆栈存储余数。输出时，从堆栈中弹出并显示余数。
3. **输出字符串**：通过DOS中断 21H 的9号功能，输出提示信息。随后通过循环 **pop dx** 从堆栈中弹出每个数字，并通过DOS中断 21H 的2号功能输出单个字符。

程序二



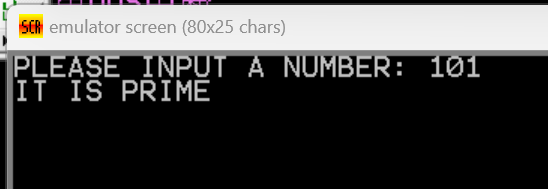
**对结果的分析**

根据输出结果，可以分析实验二的程序逻辑，确保其运作正常。这里有一些关键点：

1. **输入过程**：
   * 使用DOS中断21H的01号功能，读取10个字符，并存储在 **buf** 中。
   * 如果输出的提示信息正确，并且输入没有问题，则说明输入过程正常。
2. **最大值和最小值的计算**：
   * 使用循环比较，找出10个数字中的最大值和最小值。
   * 程序在循环中比较 **buf** 中的每个数字，通过条件判断更新最大值和最小值。
   * 如果结果显示最大值和最小值是正确的，则说明计算过程正常。
3. **输出结果**：
   * 使用DOS中断21H的09号功能，输出字符串提示信息。
   * 使用DOS中断21H的02号功能，输出最大值和最小值。
   * 如果最大值和最小值输出正确，并且与计算结果一致，则说明输出过程正常。

程序三

**输出结果**

****

输出结果主要分为两部分：

1. **输入提示**：提示用户输入一个不大于 65535 的十进制非负整数。
2. **判断结果**：判断用户输入的数是否是素数，并输出相应的字符串。

当用户输入一个十进制数后，程序会判断其是否是素数。举例来说，以下是一些可能的输入及其对应的输出结果：

* **输入 2**：这是一个素数，输出结果应该是 "It’s a prime."。
* **输入 4**：这不是素数，输出结果应该是 "It’s not a prime."。
* **输入 17**：这是一个素数，输出结果应该是 "It’s a prime."。
* **输入 20**：这不是素数，输出结果应该是 "It’s not a prime."。

**对结果的分析**

通过对实验三程序的分析，以下是对输出结果和相关逻辑的分析：

1. **输入过程**：
   * 使用 DOS 中断 21H 的 01 号功能读取用户输入的十进制数，并将其转换为整数。
   * 程序会循环读取 10 个一位数字，并通过计算来转换为整数。这一步是关键，需要确保用户输入的是十进制数字，并且正确存储在 **NUMBER** 中。
2. **判断素数**：
   * 判断逻辑通过循环来检查给定数是否是素数。
   * 素数的定义是大于 1 且只有 1 和自身两个因子的数。程序从 2 开始逐个检查，直到达到给定数的一半或找到因子。
   * 如果在判断过程中找到因子，则输出 "It’s not a prime."；否则，输出 "It’s a prime."。
3. **输出结果**：
   * 输出过程使用 DOS 中断 21H 的 09 号功能来输出字符串。
   * 程序根据判断结果输出相应的字符串。如果判断是素数，则输出 "It’s a prime."；如果判断是非素数，则输出 "It’s not a prime."。

**2.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。**

调试过程是确保汇编代码正确运行的关键环节。在上面的三个实验中，调试可能涉及程序的运行逻辑、输入输出操作、循环结构、判断逻辑、DOS中断等。以下是针对这三个实验的调试情况，包括遇到的问题及解决办法、观察到的现象及其分析。

**实验一：循环累加和的输出**

**遇到的问题**

1. **循环结构错误**：循环次数未正确设置，导致循环次数过多或不足。
2. **输出字符错误**：字符输出操作不正确，导致输出内容与预期不符。
3. **转换为十进制错误**：在累加和转换为十进制时，除法操作或堆栈操作出现问题。

**解决办法**

* **确保循环结构正确**：检查循环计数器的初始化和控制，确保循环次数正确。
* **检查输出操作**：确保使用正确的DOS中断功能，输出字符串和字符。
* **验证十进制转换**：检查除法操作和堆栈操作，确保正确转换为十进制。

**观察到的现象及其分析**

* **循环问题**：如果循环次数不对，可能会导致累加和计算错误或程序崩溃。检查循环控制，确保正确递减循环计数器。
* **字符输出问题**：如果输出操作不正确，可能会导致显示错误或乱码。确保正确使用DOS中断21H的09号和02号功能。
* **十进制转换问题**：在转换过程中，确保商和余数正确处理，避免堆栈操作错误。

**实验二：最大值和最小值的计算**

**遇到的问题**

1. **输入操作错误**：读取用户输入时，可能会遇到非法字符或意外输入。
2. **比较逻辑错误**：在循环比较中，可能因错误的判断逻辑导致计算最大值和最小值错误。
3. **输出操作错误**：输出最大值和最小值时，可能输出不正确或错位。

**解决办法**

* **确保输入正确**：检查输入字符的范围，确保只接受数字字符，避免非法字符输入。
* **验证比较逻辑**：确保比较操作和条件跳转逻辑正确，以避免判断错误。
* **检查输出操作**：确保输出最大值和最小值时，字符串和字符输出正常。

**观察到的现象及其分析**

* **输入问题**：如果输入的字符不在预期范围内，可能导致非法字符错误。确保输入范围在 '0' 到 '9'。
* **比较问题**：如果比较逻辑错误，可能导致计算最大值和最小值错误。确保比较操作正确，最大值和最小值更新逻辑正确。
* **输出问题**：如果输出操作错误，可能导致显示错误或输出错位。确保正确使用DOS中断和堆栈操作。

**实验三：判断是否是素数**

**遇到的问题**

1. **循环判断逻辑错误**：在判断是否是素数时，循环结构或条件跳转错误，可能导致判断错误。
2. **输入操作错误**：读取输入数字时，可能出现非法字符或输入错位。
3. **输出操作错误**：输出判断结果时，可能出现错误或显示乱码。

**解决办法**

* **确保循环判断逻辑正确**：检查循环结构和条件跳转，确保判断是否是素数的逻辑正确。
* **验证输入操作**：确保输入的十进制数字正确，并转换为整数。
* **检查输出操作**：确保输出素数和非素数提示时，字符串输出正确。

**观察到的现象及其分析**

* **循环判断问题**：如果判断逻辑错误，可能导致错误的素数判断。确保比较和条件跳转正确。
* **输入问题**：如果输入过程错误，可能导致无法正确读取数字或转换错误。确保输入操作和转换过程正常。
* **输出问题**：输出操作错误可能导致结果显示错误或乱码。确保正确使用DOS中断和字符串输出。

**总结**

通过对这三个实验的调试情况进行分析，可以确保汇编代码的正确性和可靠性。在调试过程中，注意观察循环结构、输入输出操作、判断逻辑、DOS中断等关键环节，以确保代码正常运行和输出结果正确。

**3.对程序设计技巧的总结及分析；实验的心得体会等。**

汇编程序设计涉及多种技巧和方法，通过对上面三个实验的总结和分析，可以获得以下几点程序设计技巧和实验心得体会。

**程序设计技巧**

1. **模块化代码**：
   * 在汇编代码中，清晰的模块化结构有助于程序的维护和扩展。通过使用空行和注释来区分代码的不同部分，有助于理解代码逻辑。
2. **循环控制**：
   * 使用循环结构时，确保循环计数器的正确初始化和更新。注意循环条件的准确性，以避免无限循环或循环次数不足的问题。
3. **数据处理**：
   * 在汇编中，数据处理通常涉及寄存器操作和内存访问。确保数据的正确存储和读取，特别是涉及堆栈和内存偏移的操作。
   * 在转换数据时，确保正确的转换逻辑，如将 ASCII 字符转换为整数或十进制转换。
4. **判断逻辑**：
   * 使用条件跳转和比较操作时，确保逻辑准确。特别是在判断是否为素数等复杂操作时，确保比较和跳转条件的正确性。
5. **输出操作**：
   * 使用 DOS 中断进行字符串和字符输出时，确保调用正确的功能号，并注意输出的内容和格式。
   * 输出操作涉及堆栈和寄存器操作，确保正确处理，以避免输出错误或乱码。
6. **结束程序**：
   * 结束程序时，使用 DOS 中断 21H 的 4C 号功能，确保程序正常结束。
   * 注意释放资源，确保堆栈和寄存器的状态恢复正常。

**实验心得体会**

1. **调试技巧**：
   * 在汇编中，调试过程至关重要。通过逐步执行、观察寄存器和内存状态，可以找出程序中的问题。
   * 调试过程中，注意循环和条件跳转的逻辑，确保代码按预期运行。
2. **数据验证**：
   * 在数据输入和处理过程中，验证数据的正确性，确保输入数据符合预期。
   * 在转换和计算过程中，确保数据转换和计算结果正确。
3. **代码清晰度**：
   * 使用注释和模块化结构，有助于提高代码的可读性和维护性。
   * 保持代码简洁，避免冗余或不必要的操作。
4. **程序设计思维**：
   * 在设计程序时，确保逻辑连贯，流程清晰。在汇编中，结构化思维和逻辑分析是关键。
   * 在实现复杂逻辑时，考虑使用子程序和模块化设计，以提高代码的可维护性。

**结论**

通过对这三个实验的总结和分析，可以看到程序设计技巧在汇编中具有重要意义。通过模块化代码、循环控制、数据处理、判断逻辑、输出操作等技巧，可以确保汇编程序的正确性和稳定性。在调试和设计过程中，注意观察和分析程序逻辑，确保代码按预期运行，输出正确结果。实验心得体会强调了调试技巧、数据验证、代码清晰度和程序设计思维的重要性。