

微机原理大作业
基于 x86 和华为服务器的模拟计算器

成员：王靳朝 周帆 陈鑫旺

指导老师：乔瑞萍老师

2023 年 5 月 17 日

目录

一.	项目要求	3
二.	基本运算式的实现	3
1.	基于 x86 实现基本运算式	3
1)	实验思路及程序流程框图	3
2)	程序源代码	4
3)	运行结果	10
2.	基于华为服务器实现基本运算式	11
1)	实验思路及程序流程框图	11
2)	程序源代码	12
3)	运行结果	12
三.	提高要求——计算器的实现	13
1.	基于 x86 的计算器实现	13
1)	实验思路及程序流程框图	13
2)	程序源代码	14
3)	执行结果	18
3.	基于华为服务器的计算器实现	18
1)	实验思路及程序流程框图	18
2)	实验环境搭建	19
3)	程序源代码	24
4)	执行结果	27

一. 项目要求

1.基本要求：实现多项式运算，如： $Z=(X+Y)\times 8\div 2$ ，其中 X,Y 可自由设定值，验证结果，给出程序和调试结果图（截屏）。

要求：用 x86 软件平台实现 $Z=(X+Y)\times 8\div 2$ 多项式功能。

2.提高要求：在“基本要求”中内容的基础上实现计算器中任意两数+、-、 \times 、 \div 功能及其扩展，可采用子程序编制加减乘除，也可采用宏命令。

3.人机交互实现：针对“提高要求”中内容，根据所选择的开发平台，完成下面的屏幕显示功能。

使用DOS功能调用增加屏幕的输入显示，输出显示功能。

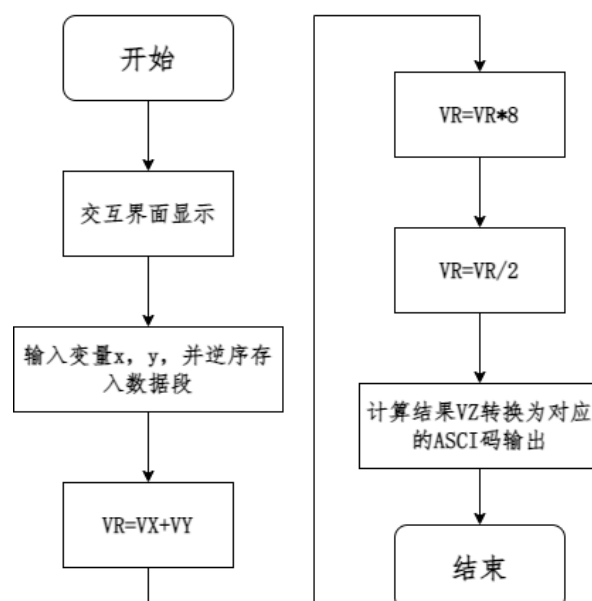
二. 基本运算式的实现

1. 基于 x86 实现基本运算式

1) 实验思路及程序流程框图

对于简单的多项式运算，程序的执行思路为顺序执行即可。首先从键盘输入 x 和 y 作为运算变量，利用出栈、入栈指令，将 x 和 y 以低位起存入内存，一字节存入一位，最高十位。从低位到高位按字节进行 $x+y$ 和 $\times 8$ 的运算，并将运算结果存入中间变量；从高位到低位按字节进行 $\div 2$ 运算，并将计算结果从高位到低位依字节存入结果变量。最后将运算结果转换为对应的 ASCII 码，利用 DOS 功能调用显示在屏幕上。并设计简单的 UI 使人机交互更加清晰简便。

程序框图如下：



2) 程序源代码

```
;-----  
;程序描述：  
;  可实现多项式 $Z=(X+Y)*8/2$ 在10位数以内的运算，不考虑除法溢出问题  
;作者：周帆  
;时间：2023年5月13日  
;输入格式样例：  
;  运行程序后，从键盘输入X的数值X=12，按下回车之后，从键盘输入Y的  
数值Y=34  
;  系统运算得到运算式的结果并显示在屏幕上  
;调试工具：  
;  HQFC-A微机接口  
;说明：无  
;-----  
CRLF      MACRO                ;显示回车和换行  
    MOV AH,02H  
    MOV DL,0DH  
    INT 21H  
    MOV AH,02H  
    MOV DL,0AH  
    INT 21H  
    ENDM  
STACK1SEGMENT  
    DB 256 DUP(?)  
STACK1ENDS  
DATA      SEGMENT  
    VX DB 10 DUP(?)            ;预置变量X,Y,Z的存储空间  
    VY DB 10 DUP(?)  
    VR DB 12 DUP(?)            ;中间变量  
    VZ DB 13 DUP(?)  
    SSE DB 50 DUP('+'),0DH,0AH,'$';交互界面分隔  
    SPO DB 'Polynomial:Z=(X+Y)*8/2',0DH,0AH,'$';提示用户默认的多项  
式运算  
    SVX DB 'X=',0DH,0AH,'$'    ;提示用户输入X  
    SVY DB 'Y=',0DH,0AH,'$'    ;提示用户输入Y  
    SVZ DB 'Z=',0DH,0AH,'$'    ;输入结果Z  
DATA      ENDS  
CODE      SEGMENT
```

```

    ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA          ;数据段
        MOV DS,AX
        MOV AX,STACK1      ;栈段
        MOV SS,AX
        MOV AH,09H         ;显示分割线
        LEA DX,SSE
        INT 21H
        MOV AH,09H         ;显示运算式
        LEA DX,SPO
        INT 21H
        ;输入变量X,并逆序存入数据段
        MOV AH,09H         ;提示用户输入变量X
        LEA DX,SVX
        INT 21H
        MOV BX,0           ;统计字节长
        MOV CX,10          ;设置循环次数上限
        LEA SI,VX          ;指针指向变量存储首地址
LIX1:   MOV AH,08H         ;逐字输入
        INT 21H
        CMP AL,0DH         ;检测，输入回车符表示输入完成
        JZ JIENDX          ;完成，跳转
        MOV DL,AL          ;显示
        MOV AH,02H
        INT 21H
        SUB AL,30H         ;将ASCII码转换为十进制数，方便运算
        PUSH AX            ;入栈
        INC BX             ;计数
        LOOP LIX1
JIENDX: CRLF
        MOV CX,BX          ;设置循环次数为字节长
LIX2:   POP AX             ;出栈
        MOV [SI],AL        ;未完成，则将字符存入
        INC SI             ;指针步进
        LOOP LIX2
        ;输入变量Y,并逆序存入数据段
        MOV AH,09H         ;提示用户输入变量Y
        LEA DX,SVY

```

```

INT 21H
MOV BX,0           ;统计字节长
MOV CX,10          ;设置循环次数上限
LEA SI,VY          ;指针指向变量存储首地址
LIY1:  MOV AH,08H   ;逐字输入
INT 21H
CMP AL,0DH         ;检测，输入回车符表示输入完成
JZ JIENDY          ;完成，跳转
MOV DL,AL          ;显示
MOV AH,02H
INT 21H
SUB AL,30H         ;将ASCII码转换为十进制数，方便运算
PUSH AX            ;入栈
INC BX             ;计数
LOOP LIY1
JIENDY: CRLF
MOV CX,BX          ;设置循环次数为字节长
LIY2:  POP AX       ;出栈
MOV [SI],AL        ;未完成，则将字符存入
INC SI             ;指针步进
LOOP LIY2
;计算及输出
CALL CADD          ;加法计算
CALL CMUL          ;乘法计算
CALL CDIV          ;除法计算
CALL TRANS         ;结果转换为ASCII码
CALL ORESULT       ;输出计算结果
MOV AH,4CH
INT 21H

CADD  PROC NEAR    ;加法计算子程序
PUSH SI           ;保护现场
PUSH DI
PUSH BX
PUSH CX
PUSH AX
LEA SI,VX         ;指针SI指向变量X首地址
LEA DI,VY         ;指针DI指向变量Y首地址

```

	LEA BX,VR	;指针BX指向变量R首地址
	MOV CX,10	;设置循环次数
	XOR AX,AX	;AX清零
	CLC	
LADD:	MOV AL,[SI]	;逐位相加
	ADC AL,[DI]	
	AAA	
	MOV [BX],AL	;存入结果
	INC SI	;指针步进
	INC DI	
	INC BX	
	LOOP LADD	
	JNC JEND	
	MOV [BX],01H	;进位计入
JEND:	POP AX	;恢复现场
	POP CX	
	POP BX	
	POP DI	
	POP SI	
	RET	
CADD	ENDP	
CMUL	PROC NEAR	;乘法计算子程序
	PUSH DX	;保护现场
	PUSH SI	
	PUSH BX	
	PUSH CX	
	PUSH AX	
	MOV DL,08H	;乘数
	LEA SI,VR	;指针SI指向变量R首地址
	MOV BL,00H	;存放进位
	MOV CX,12	;设置循环次数
	CLC	
LMUL:	MOV AL,[SI]	
	MUL DL	;逐位相乘
	AAM	
	ADD AL,BL	;加入进位
	AAA	

	MOV [SI],AL	;结果低四位存入AL
	MOV BL,AH	;进位存入BL
	XOR AX,AX	;AX清零
	INC SI	;指针步进
	LOOP LMUL	
	MOV [SI],BL	;存入进位
	POP AX	;恢复现场
	POP CX	
	POP BX	
	POP SI	
	POP DX	
	RET	
CMUL	ENDP	
CDIV	PROC NEAR	;除法计算子程序
	PUSH DX	;保护现场
	PUSH DI	
	PUSH SI	
	PUSH AX	
	PUSH CX	
	MOV DL,02H	;存入除数
	LEA DI,VZ	;指针DI指向变量Z首地址
	LEA SI,VZ	;指针SI指向变量R末地址
	DEC SI	
	MOV AH,[SI]	;AH装入
	MOV AL,00H	;AL清零
	DEC SI	
	INC DI	
	MOV CX,11	;设置循环次数(将第0、1位装入AX，避免除法溢出)
	CLC	
LDIV:	MOV AL,[SI]	;取出变量Z的字节
	AAD	
	DIV DL	;逐位相除
	MOV [DI],AL	;结果低四位存入AL
	XOR AL,AL	;AL清零
	DEC SI	;指针步退
	INC DI	;指针步进


```

        LOOP LDIV
        POP CX           ;恢复现场
        POP AX
        POP SI
        POP DI
        POP DX
        RET
CDIV     ENDP

TRANS    PROC NEAR           ;将计算结果转换为ASCII码
        PUSH SI           ;保护现场
        PUSH AX
        LEA SI,VZ         ;指针指向存储首地址
        MOV CX,12         ;设置循环次数
LTRANS:  MOV AL,[SI]
        ADD AL,30H
        MOV [SI],AL
        INC SI           ;指针步进
        LOOP LTRANS
        MOV [SI],'$'     ;字符串结束标识符存入
        POP AX           ;恢复现场
        POP SI
        RET
TRANS    ENDP

ORESULT  PROC NEAR           ;将计算结果输出到屏幕
        PUSH AX           ;保护现场
        PUSH DX
        MOV AH,09H        ;提示输出结果
        LEA DX,SVZ
        INT 21H
        MOV CX,12         ;设置循环次数
        LEA SI,VZ         ;指针指向结果Z
LO:      MOV DL,[SI]
        MOV AH,02H        ;输出结果
        INT 21H
        INC SI           ;指针步进
        LOOP LO

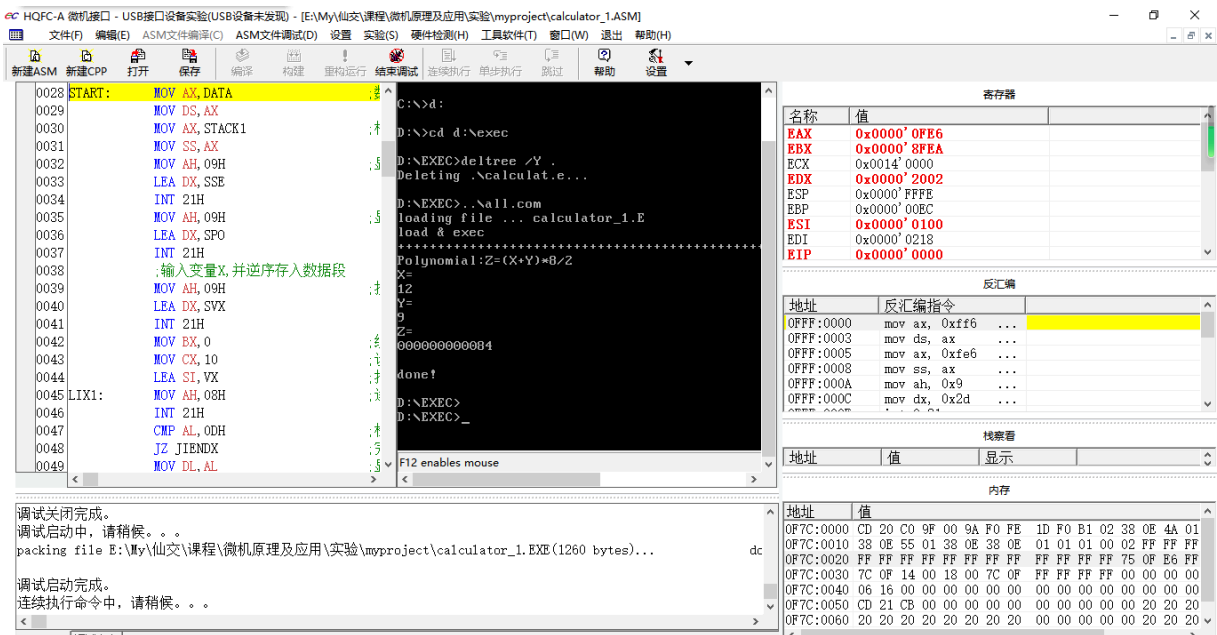
```

```
CRLF
POP DX          ;恢复现场
POP AX
RET
ORESULT  ENDP

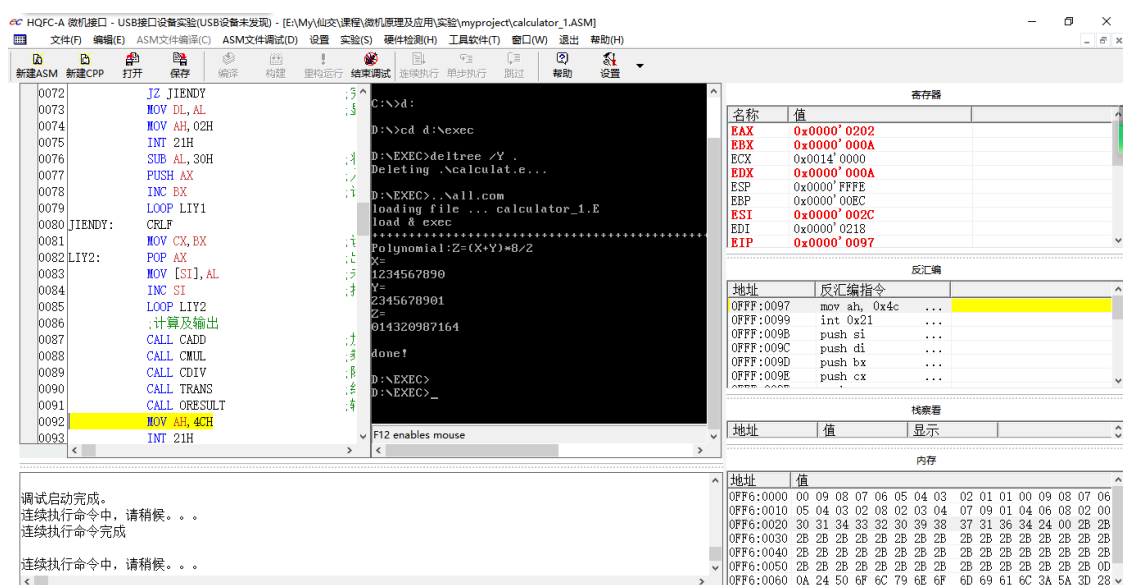
CODE           ENDS
END START
```

3) 运行结果

X=12,Y=9,得到 Z=84



X=1234567890,Y=2345678901,得到 Z=14320987164

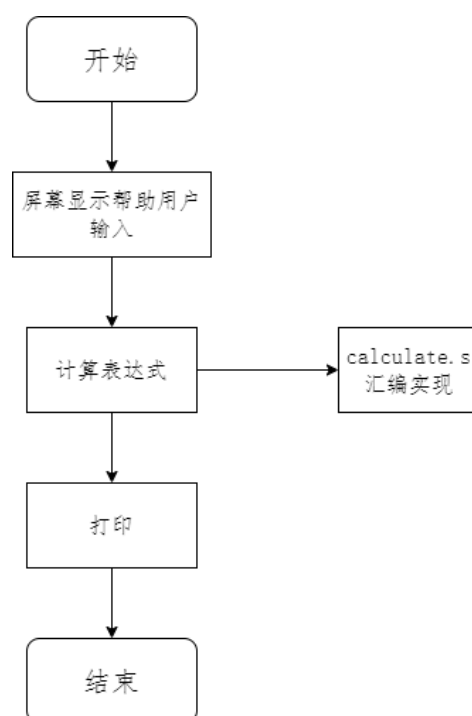


2. 基于华为服务器实现基本运算式

利用 C 语言和 ARM-V8 混合语言编程的总体思路不变，可以使用 C 语言实现输出变量，并实现在屏幕上显示，可以直接使用 C 语言自带的运算符或者汇编语言实现多项式运算。由于本次着重训练软件接口技术，因此使用 C 语言和 ARM-V8 汇编语言混合编程实现多项式运算。

1) 实验思路及程序流程图

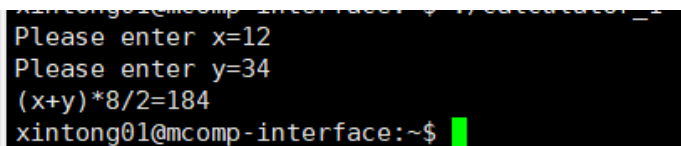
总体思路和 x86 环境下的程序不变，程序框图为：



2) 程序源代码

```
//-----  
//程序描述：实现 $(x+y)*8/2$  运算式，定义输入输出均为 int 类型  
//作者：王靳朝  
//时间：2023 年 5 月 16 日  
//输入格式样例：  
//      运行程序后，从键盘输入 x=12，y=34，回车之后程序运行  
//      将结果显示在屏幕上  
//调试工具：Xshell7 华为服务器  
//说明：无  
//-----  
#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
    int x,y,z;  
    printf("Please enter x=");  
    scanf("%d",&x);  
    printf("Please enter y=");  
    scanf("%d",&y);  
    printf("(x+y)*8/2=%d\n",calculate(x,y));  
    return 0;  
}  
  
.global calculate  
    mov w0,#0  
calculate:  
    add w0,w1,w0  
    lsl w0,w0,3  
    lsr w0,w0,1  
    ret
```

3) 运行结果



```
xintong01@mcomp-interface:~/calcuator_1  
Please enter x=12  
Please enter y=34  
(x+y)*8/2=184  
xintong01@mcomp-interface:~$
```

```
xintong01@mcomp-interface: ~$ ./calculator
Please enter x=3
Please enter y=9
(x+y)*8/2=48
xintong01@mcomp-interface: ~$
```

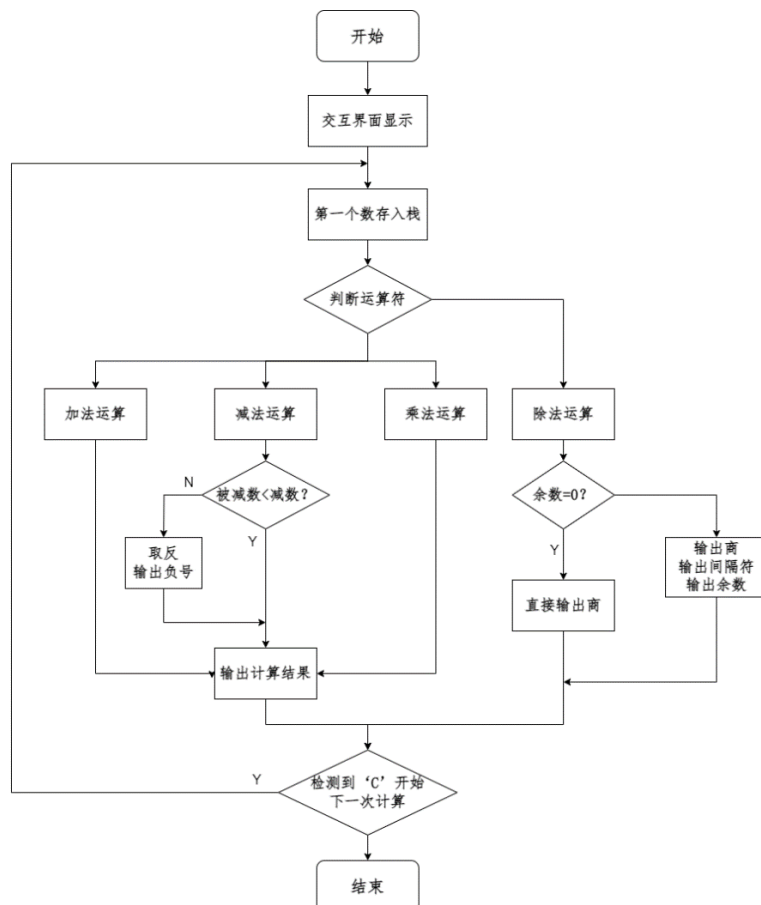
```
xintong01@mcomp-interface: ~$ ./calculator
Please enter x=100
Please enter y=9522
(x+y)*8/2=38488
xintong01@mcomp-interface: ~$
```

三. 提高要求——计算器的实现

1. 基于 x86 的计算器实现

1) 实验思路及程序流程框图

开始运行程序之后首先显示交互界面，将第一个数存入堆栈，用户输入运算符进行判断，对于不同的运算符进行不同的操作，特别的对于减法运算需要判断结果是否为负数，对于除法运算需要判断是否存在余数，接着输入第二个数，存入寄存器，用户按下 '=' 运算并输出结果。最后检测用户是否按下 'C'，如果按下，则模拟计算器清 0 开始下一次运算，否则结束。



2) 程序源代码

```
;-----  
;程序描述:  
; 通过入栈/出栈操作记录输入数据,实现两个数之间的加、减、乘、除运  
算,其中  
; 加法:两位加数、和范围均为:0~65536  
; 减法:被减数、减数、差范围均为:-32767~32767  
; 乘法:两位乘数、积范围均为:0~65536  
; 除法:被除数、除数、商、余数范围均为:0~32767  
; '='--输出结果,'C'--归零  
;作者:周帆  
;日期:2023年5月13日  
;输入格式样例:  
; 输入12+34=,自动计算结果并显示在运算时之后。结果自动显示46.  
; 进行下一次计算时,按下键盘C键确认模拟计算器清0,开始下一次计算  
;调试工具:  
; HQFC-A微机接口  
;说明:无  
;-----  
CRLF          MACRO                ;显示回车和换行  
    MOV AH,02H  
    MOV DL,0DH  
    INT 21H  
    MOV AH,02H  
    MOV DL,0AH  
    INT 21H  
    ENDM  
STACK1 SEGMENT  
    DB 256 DUP(?)  
STACK1 ENDS  
DATA     SEGMENT  
    RE DB 0,0,0,0,0,'$'      ;计算结果的ASCII码  
    DIVI DW 10000,1000,100,10,1 ;万、千、百、十、个  
    SE DB 6 DUP('.),'$'      ;除法间隔符  
    SSE DB 50 DUP('+'),0DH,0AH,'$';交互界面分隔  
    STIP DB 'My Calculator:',0DH,0AH,'$';提示用户输入计算式  
DATA     ENDS  
CODE     SEGMENT
```

```

    ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA           ;数据段
        MOV DS,AX
        MOV AX,STACK1       ;栈段
        MOV SS,AX
        MOV AH,09H          ;显示分割线
        LEA DX,SSE
        INT 21H
        MOV AH,09H          ;显示提示词
        LEA DX,STIP
        INT 21H
REFRESH: XOR AX,AX           ;AX清零
        CALL INPUT           ;输入第一个数
        PUSH BX              ;存入栈
        ;判断运算符，跳转
        CMP AL,'+'
        JZ CADD
        CMP AL,'-'
        JZ CSUB
        CMP AL,'*'
        JZ CMUL
        CMP AL,'/'
        JZ CDIV
CADD:   CALL INPUT           ;输入第二个数
        POP AX               ;第一个数取出
        ADD AX,BX            ;相加
        JMP NEXT            ;跳出
CSUB:   CALL INPUT           ;输入第二个数
        POP AX               ;第一个数取出
        CMP AX,BX            ;判断被减数与减数的大小关系
        JL LESS              ;小于，跳转
        SUB AX,BX            ;大于，直接相减
        JMP NEXT
LESS:   SUB AX,BX            ;相减为负则取反
        NEG AX
        PUSH AX              ;保护AX
        MOV DL,'-'          ;输出负号
        MOV AH,02H

```

```

INT 21H
POP AX          ;送回AX
JMP NEXT        ;跳出
CMUL: CALL INPUT          ;输入第二个数
POP AX          ;第一个数取出
MUL BX          ;相乘
JMP NEXT        ;跳出
CDIV: CALL INPUT          ;输入第二个数
POP AX          ;第一个数取出
DIV BX          ;相除
CMP DX,0        ;检测余数
JZ NEXT         ;直接跳出
PUSH DX         ;保护DX
CALL OUTPUT     ;输出商
MOV AH,09H      ;显示间隔符
LEA DX,SE
INT 21H
POP DX          ;送回DX
MOV AX,DX       ;输出余数
CALL OUTPUT
JMP REFRESH     ;跳出，刷新
NEXT: CALL OUTPUT          ;结果输出
JMP REFRESH     ;跳出，刷新
MOV AH,4CH
INT 21H

INPUT PROC NEAR          ;输入子程序
MOV BX,0              ;记录已输入的数值，初始化BX清零
INNUM: MOV AH,01H      ;字符输入
INT 21H
CMP AL,'C'            ;归零跳转
JZ CLEAR
CMP AL,'+'            ;运算符跳出
JZ EXIT
CMP AL,'-'
JZ EXIT
CMP AL,'*'
JZ EXIT

```



```

    CMP AL','
    JZ EXIT
    SUB AL,30H           ;ASCII码转换为十进制
    JL EXIT             ;排除0~9以外的字符
    CMP AL,9
    JG EXIT
    CBW                 ;按数位依次将输入的运算数存入
    XCHG AX,BX          ;已输入数值送入
    MOV CX,10           ;十进制
    MUL CX              ;十进制下左移一位
    XCHG AX,BX          ;送回
    ADD BX,AX           ;当前输入位加入
    JMP INNUM           ;跳转输入下一位
CLEAR: CRLF            ;回车换行
    XOR AX,AX           ;寄存器AX,BX清零
    XOR BX,BX
    JMP INNUM           ;跳转输入下一位
EXIT:    RET
INPUT    ENDP

OUTPUT   PROC NEAR     ;输出子程序
    MOV SI,OFFSET RE   ;指针
    MOV DI,OFFSET DIVI
    MOV CX,5           ;设置循环次数
LTRANS:  MOV DX,0       ;DX清零,存储未转换数值
    DIV WORD PTR [DI]   ;位数
    ADD AL,30H         ;十进制转换为ASCII码
    MOV BYTE PTR [SI],AL ;存储
    INC SI             ;指针步进
    ADD DI,2
    MOV AX,DX          ;未转换值送回
    LOOP LTRANS
    MOV SI,OFFSET RE   ;指针
    MOV CX,4           ;设置循环次数
LPRINT:  CMP BYTE PTR [SI],30H ;从第一个有效位开始输出
    JNZ PRINT
    INC SI             ;指针步进
    LOOP LPRINT

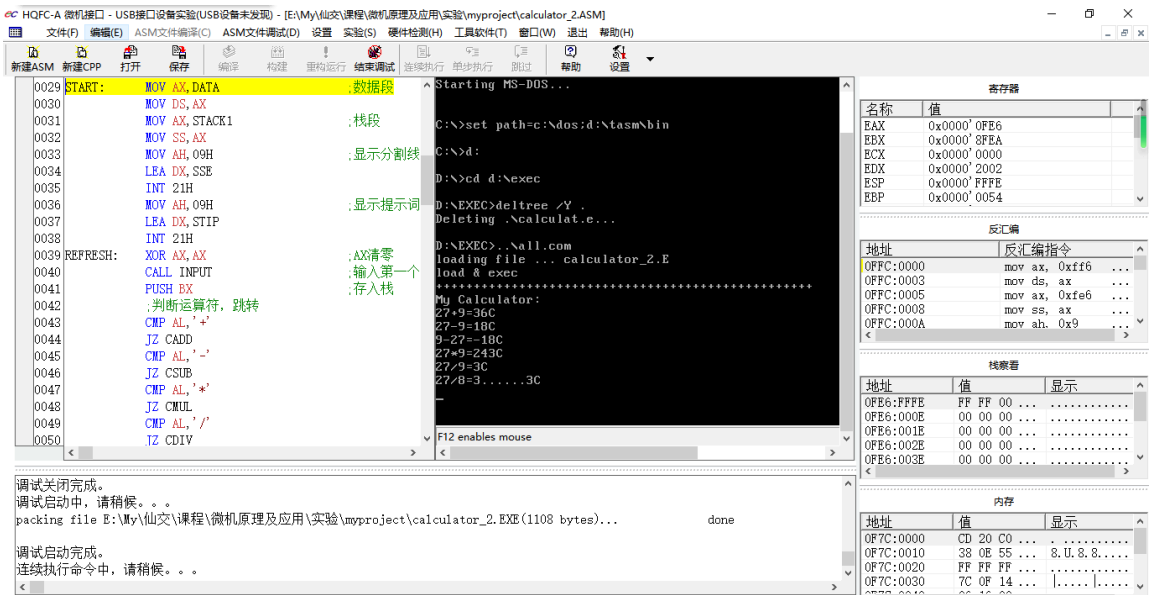
```

```
PRINT: MOV DX,SI ;计算结果输出
      MOV AH,09H
      INT 21H
      RET
OUTPUT  ENDP

CODE  ENDS
      END START
```

3) 执行结果

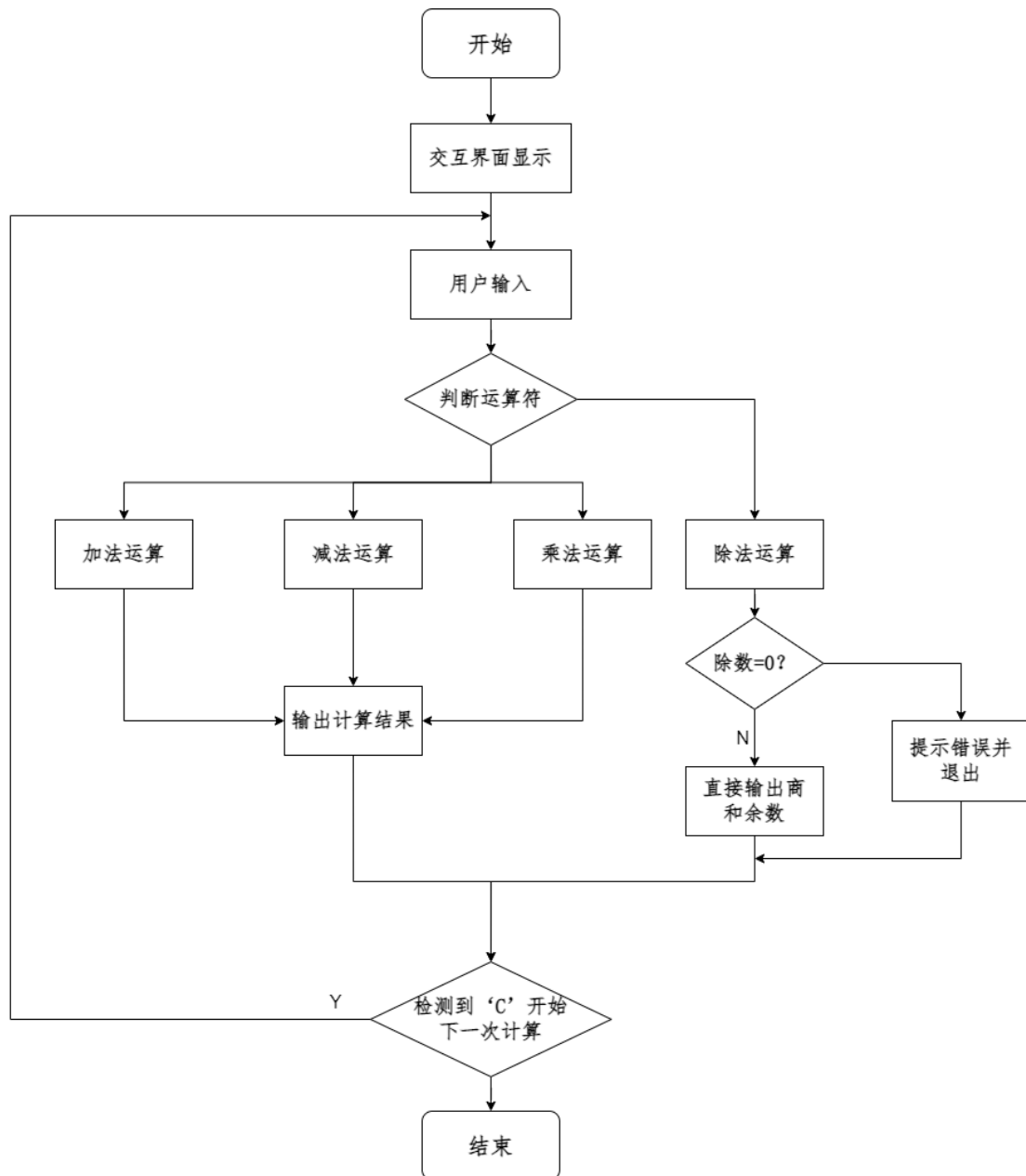
可以观察到，输入 27+9 输出 36，键盘输入 C 之后进行下一次计算，对于除法运算 27/9，整除直接得出 3，对于无法整除 27/8，显示结果 3.....3，前者为商，后者为余数，表示商为 3，余数也为 3。



3. 基于华为服务器的计算器实现

1) 实验思路及程序流程框图

华为服务器下使用 C 语言和 ARM-V8 混合语言编程。利用 C 语言可以构建基本的 UI 环境和输入输出函数，计算函数利用汇编语言实现。当程序开始运行之后首先显示 UI 界面，根据用户输入进行不同的操作，对错误的操作需要进行提示和退出。注意 C 语言和汇编语言的接口。每一次运算结束之后从键盘输入，检测是否为'C'，如果是，则继续下一次运算，从头开始运行，如果不是则退出计算器，因此使用 do-while 语句实现上述功能。程序框图如下：



2) 实验环境搭建

参照相关文档将 Xshell7 和 Xftp7 安装完成后，两个软件快捷方式在统一文件夹下：

查看

ell

在 Xshell 中搜索

名称	修改日期	类型	大小
Xftp 7	2023/5/16 17:51	快捷方式	1 KB
Xshell 7	2023/5/15 21:32	快捷方式	1 KB

桌面快捷方式文件夹中新建 startXShell.txt 文本文件，并在其中输入以下：

```
#####begin#####
###
@echo off
%1 mshta
vbscript:CreateObject("Shell.Application").ShellExecute("cmd.exe","/c%~s0::",
"", "runas",1)(window.close)
title Xshell 启动器
set atime=%date:~0,4%-~5,2%-~8,2%

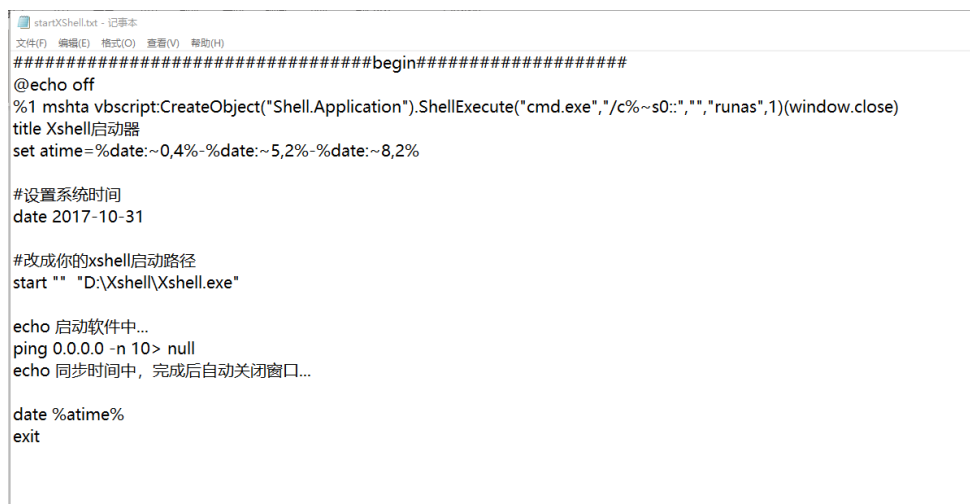
#设置系统时间
date 2017-10-31

#改成你的 xshell 启动路径
start "" "D:\Xshell\Xshell.exe"

echo 启动软件中...
ping 0.0.0.0 -n 10> null
echo 同步时间中，完成后自动关闭窗口...

date %atime%
exit
```

将其中 Xshell 的路径更改为自己的安装路径



```
startXShell.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#####begin#####
@echo off
%1 mshta vbscript:CreateObject("Shell.Application").ShellExecute("cmd.exe","/c%~s0::", "", "runas",1)(window.close)
title Xshell启动器
set atime=%date:~0,4%-~5,2%-~8,2%

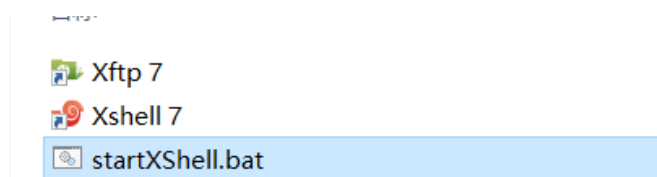
#设置系统时间
date 2017-10-31

#改成你的xshell启动路径
start "" "D:\Xshell\Xshell.exe"

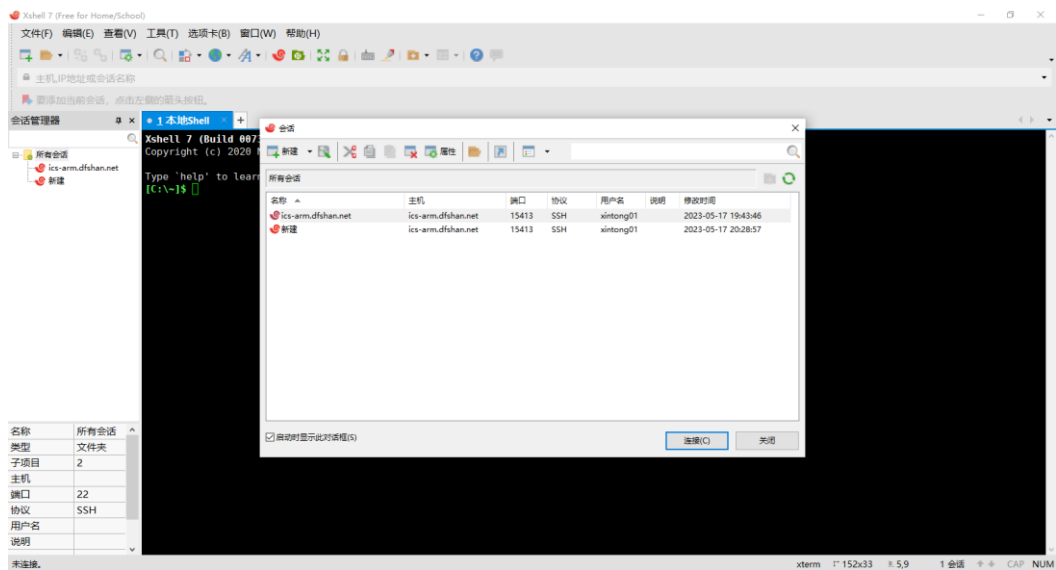
echo 启动软件中...
ping 0.0.0.0 -n 10> null
echo 同步时间中，完成后自动关闭窗口...

date %atime%
exit
```

编辑完成后将文件后缀改为.bat



以管理员身份运行，即可以正常进入 Xshell7 软件：

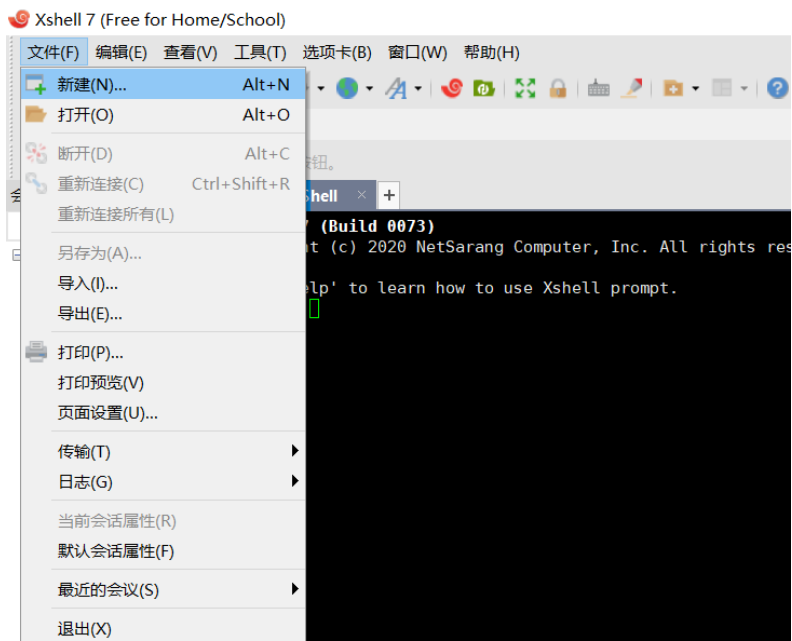


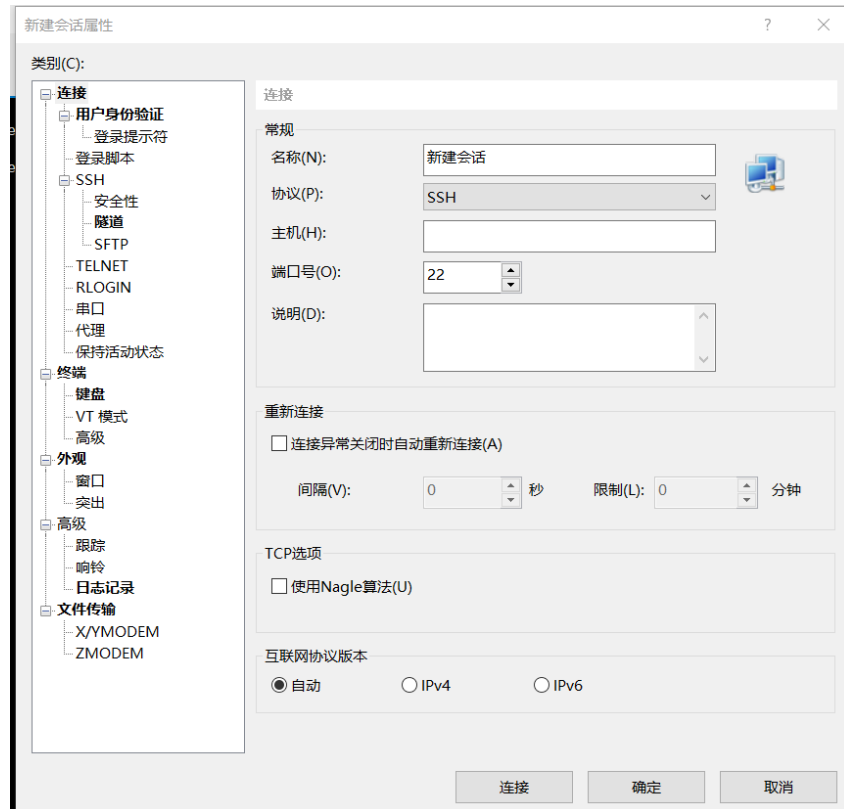
如果有文件管理需求，需要使用 Xftp7，同样会出现需要更新的问题，解决方法类似，可以更改系统时间，更推荐使用方法 2。代码部分只需将【Xshell7】的安装路径】更改为【Xftp7】的安装路径即可，文件命名可根据自己喜好区分，其他完全相同。

因此快捷方式中共以下文件：

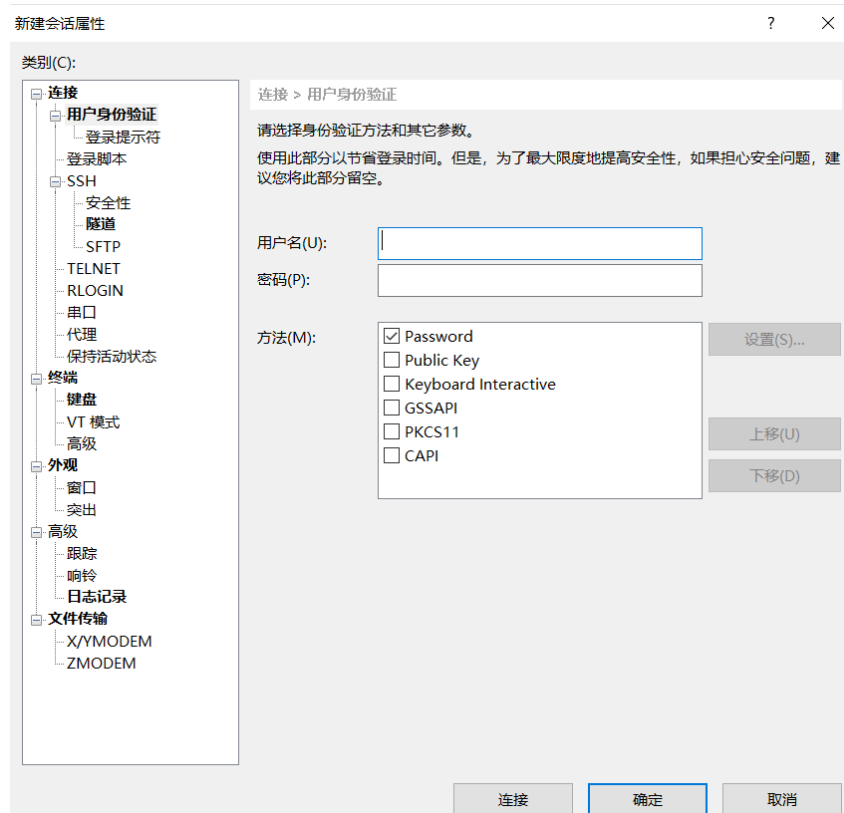
名称	修改日期	类型	大小
Xftp 7	2023/5/16 17:51	快捷方式	1 KB
Xshell 7	2023/5/15 21:32	快捷方式	1 KB
startXShell.bat	2022/10/1 16:58	Windows 批...	1 KB
startXftp.bat	2023/5/17 21:01	Windows 批...	1 KB

在 Xshell 主界面，点击左上角【文件】-【新建】，弹出新建会话属性界面：





将老师提供的主机名填入，并更改端口号，协议为 SSH 不用更改。之后点击左上角【用户身份验证】：



其中用户名：xintong01，password：xintong01
输入老师提供的用户名及密码，确认无误后点击连接，即可成功连接华为服务器，

此时 Xshell 界面显示为:

会话管理器

所有会话

ics-arm.dfshan.net

新建

名称	ics-arm.dfs...	56 updates can be applied immediately.
主机	ics-arm.dfs...	To see these additional updates run: apt list --upgradable
端口	15413	Enable ESM Apps to receive additional future security updates.
协议	SSH	See https://ubuntu.com/esm or run: sudo pro status
用户名	xintong01	
说明		

```
Host 'ics-arm.dfshan.net' resolved to 10.181.8.149.
Connecting to 10.181.8.149:15413...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.

Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-69-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Mon May 29 10:11:57 AM UTC 2023

System load:  0.0               Processes:           210
Usage of /:   21.4% of 95.11GB   Users logged in:    0
Memory usage: 3%                IPv4 address for enp1s0: 192.168.122.196
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

 * Introducing Expanded Security Maintenance for Applications.
Receive updates to over 25,000 software packages with your
Ubuntu Pro subscription. Free for personal use.

https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

*** System restart required ***
Last login: Thu May 18 07:49:45 2023 from 192.168.122.1
xintong01@mcomp-interface:~$
```

左侧显示会话，可以进行管理，左下显示相关信息，屏幕显示服务器的使用情况等。

建立计算器主函数文件 calculato_2, 以及子函数文件 add.s、sub.s、mul.s、div.s:

```
xintong01@mcomp-interface:~$
xintong01@mcomp-interface:~$ nano calculator_2.c
xintong01@mcomp-interface:~$ nano add.s
xintong01@mcomp-interface:~$ nano sub.s
xintong01@mcomp-interface:~$ nano mul.s
xintong01@mcomp-interface:~$ nano div.s
xintong01@mcomp-interface:~$
```

1 ics-arm.dfshan.net

GNU nano 6.2

```
global add
mov w0,#0

add:
add w0,w0,w1
ret
```

1 ics-arm.dfshan.net

GNU nano 6.2

```
global sub
mov w0,#0

sub:
sub w0,w0,w1
ret
```

1 ics-arm.dfshan.net

GNU nano 6.2

```
global mul
mov w0,#0

mul:
mul w0,w0,w1
ret
```

1 ics-arm.dfshan.net

GNU nano 6.2

```
global div
mov w0,#0

div:
sdiv w0,w0,w1
ret
```

```

GNU nano 6.2
//-----
//程序描述：可以实现任意两个数的加减乘除运算，输入输出
//      均为int类型
//作者：王靳朝
//时间：2023年5月16日
//输入样例：UI界面显示之后，用户根据提示选择需要操作的类型
//      例如，选择4号为除法运算，根据提示输入x=7, y=3
//      运行结果之后显示商为2，余数为1.
//      用户按下C之后模拟计算器清0，开始重新运算
//调试工具：Xshell7 华为服务器
//说明：无
//-----
#include<stdio.h>
#include<stdlib.h>
void menu()
{
    printf("|-----计算器-----|\n");
    printf("|-----*1.Add*-----|\n");
    printf("|-----*2.Sub*-----|\n");
    printf("|-----*3.Mul*-----|\n");
    printf("|-----*4.Div*-----|\n");
    printf("|-----*0.Exit*-----|\n");
}

int  add(int w0,int w1);
int  sub(int w0,int w1);
int  mul(int w0,int w1);
extern div_t  div(int w0,int w1);

```

(完整代码见第 3)部分)

完成编辑后将代码编译并生成可执行文件，之后运行：

```

xintong01@mcomp-interface:~$
xintong01@mcomp-interface:~$ gcc calculator_2.c add.s sub.s mul.s div.s -o calculator_2
calculator_2.c: In function 'main':
calculator_2.c:63:73: warning: format '%d' expects argument of type 'int', but argument 2 has type 'div_t' [-Wformat=]
   63 |         printf("The quotient of x/y is %d\n",div(x,y));
      |                                     ^~      ~~~~~
      |                                     |      |
      |                                     int  div_t
xintong01@mcomp-interface:~$ ./calculator_2
|-----计算器-----|
|-----*1.Add*-----|
|-----*2.Sub*-----|
|-----*3.Mul*-----|
|-----*4.Div*-----|
|-----*0.Exit*-----|
Please select the operation type

```

由于我们规定除法输出类型为整数型，但是除法运算结果可能并非整数型，因此会出现警告。

3) 程序源代码

```

//-----
//程序描述：可以实现任意两个数的加减乘除运算，输入输出
//      均为 int 类型
//作者：王靳朝
//时间：2023 年 5 月 16 日

```



```

//输入样例：UI 界面显示之后，用户根据提示选择需要操作的类型
//      例如，选择 4 号为除法运算，根据提示输入 x=7，y=3
//      运行结果之后显示商为 2，余数为 1.
//      用户按下 C 之后模拟计算器清 0，开始重新运算
//调试工具：Xshell7 华为服务器
//说明：无
//-----
#include<stdio.h>
#include<stdlib.h>
void menu()
{
    printf("|-----计算器-----|\n");
    printf("|-----*1.Add*-----|\n");
    printf("|-----*2.Sub*-----|\n");
    printf("|-----*3.Mul*-----|\n");
    printf("|-----*4.Div*-----|\n");
    printf("|-----*0.Exit*-----|\n");
}

int  add(int w0,int w1);
int  sub(int w0,int w1);
int  mul(int w0,int w1);
extern div_t  div(int w0,int w1);

int main()
{
    int x,y,z,a;
    char c;
    menu();
    do
    {
        printf("Please select the operation type\n");
        scanf("%d",&a);
        if(a == 0)
        {
            printf("Exit\n");
            return 0;
        }
    }
}

```

```

printf("Please enter the number of operands\nx=");
scanf("%d",&x);
printf("y=");
scanf("%d",&y);

switch (a)
{
case 1:
    printf("x+y=%d\n",add(x,y));
    break;
case 2:
    printf("x-y=%d\n",sub(x,y));
    break;
case 3:
    printf("x*y=%d\n",mul(x,y));
    break;
case 4:
    if (y != 0)
    {
        printf("The quotient of x/y is %d\n",div(x,y));
        printf("The remainder of x/y is %d\n",x-(x/y)*y);
        break;
    }
    else
    {
        printf("Error!Please check the number of operands!\n");
    }
    default:
        printf("Invaild operation\n");
        break;
}
printf("Please Press 'C' to clear,press others to exit:\n");
scanf(" %c",&c);
}while(c == 'C');
return 0;
}

add.s:

```

```

.global add
    mov w0,#0
add:
    add w0,w0,w1
    ret

sub.s:
.global sub
    mov w0,#0
sub:
    sub w0,w0,w1
    ret

mul.s:
.global mul
    mov w0,#0
mul:
    mul w0,w0,w1
    ret

div.s:
.global div
    mov w0,#0
div:
    sdiv w0,w0,w1
    ret

```

4) 执行结果

首先测试退出功能：交互界面选择 0 号功能可以正常退出

```

xintong01@mcomp-interface:~$ ./calculator_2
|-----计算器-----|
|-----*1.Add*-----|
|-----*2.Sub*-----|
|-----*3.Mul*-----|
|-----*4.Div*-----|
|-----*0.Exit*-----|
Please select the operation type
0
Exit

```

其次测试加法和减法功能以及能否正常开启下一次运算，选择加法和减法功能，

检查发现结果均正确，同时运算结束之后键盘输入 C 键可以正常开启下一次运算。下一次运算需要重新选择操作类型并重新输入操作数。

```
EXIT
xintong01@mcomp-interface:~$ ./calculator_2
|-----计算器-----|
|-----*1.Add*-----|
|-----*2.Sub*-----|
|-----*3.Mul*-----|
|-----*4.Div*-----|
|-----*0.Exit*-----|
Please select the operation type
1
Please enter the number of operands
x=-5
y=050
x+y=45
Please Press 'C' to clear,press others to exit:
C
Please select the operation type
2
Please enter the number of operands
x=90
y=1088
x-y=-998
Please Press 'C' to clear,press others to exit:
```

检测运算符不合法的情况，可以正常提示并退出。

```
C
Please select the operation type
31
Please enter the number of operands
x=5
y=6
Invaild operation
```

乘法运算可以得出正确结果：

```
C
Please select the operation type
3
Please enter the number of operands
x=5
y=0
x*y=0
Please Press 'C' to clear,press others to exit:
C
```

对于整除和非整除情况均可以得到正常结果并分别显示商和余数：

```
C
Please select the operation type
4
Please enter the number of operands
x=64
y=8
The quotient of x/y is 8
The remainder of x/y is 0
Please Press 'C' to clear,press others to exit:
C
Please select the operation type
4
Please enter the number of operands
x=64
y=7
The quotient of x/y is 9
The remainder of x/y is 1
Please Press 'C' to clear,press others to exit:
```

若除数为 0，则可以正常提示错误信息。运算结束之后按下非'C'按键可以正常退出。

```
C
Please select the operation type
4
Please enter the number of operands
x=18
y=0
Error!Please check the number of operands!
Invaild operation
Please Press 'C' to clear,press others to exit:
g
xintong01@mcomp-interface:~$
```