目录

一、多项式实现

1. x86 实现（x+y）*4/2

（1）设计思路

由 add 实现加法，shl 实现乘法，shr 实现除法。

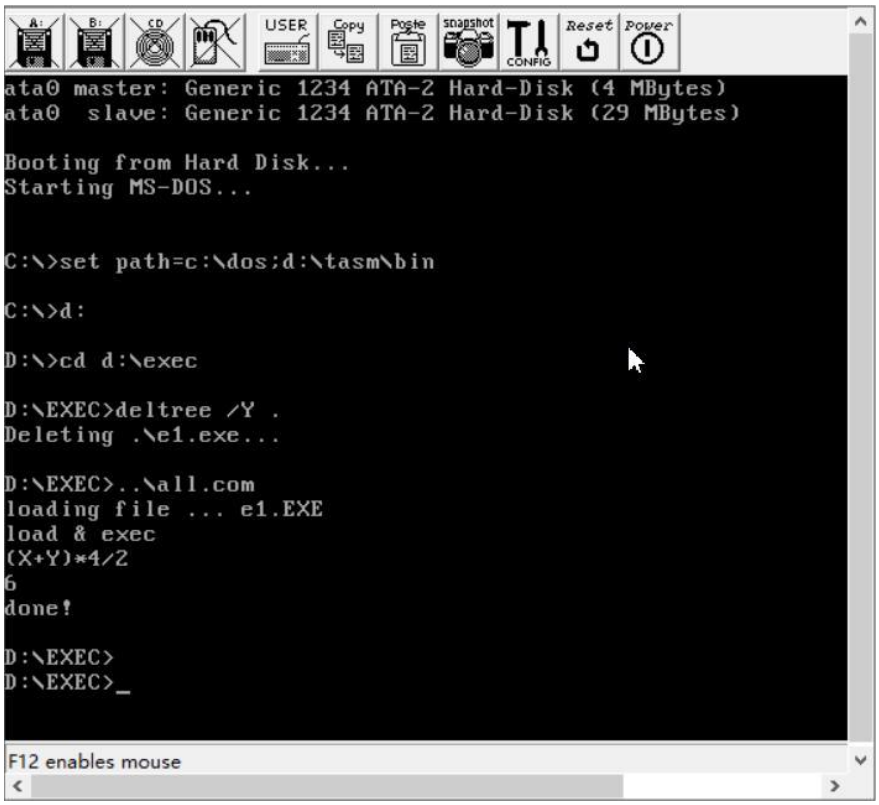
（2）源代码

```
DATA SEGMENT
   X DB 01H
   Y DB 02H
   STRING DB '(X+Y)*4/2',0DH,0AH,'$'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
   MOV DS,AX
   LEA DX,STRING
   MOV AH,09H
   INT 21H
   XOR AX,AX
   MOV AL,X
   ADD AL,Y
   JC NEXT
   JMP NEXT1
```

```
NEXT: ADD AH,01H
NEXT1:SHL AX,2
   SHR AX,1
   CMP AL,09H
   JA NEXT2
   ADD AL,30H
   JMP NEXT3
NEXT2: ADD AL,37H
NEXT3:

   XOR DX,DX
   MOV DX,AX
   MOV AH,02H
   INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

（3）运行结果

x=1,y=1　　输出结果为 6



2. 华为云基于鲲鹏处理器 ARM V8 软件平台实现

（1）设计思路

c 语言与 ArmV8 汇编混合编程（（x+y）*8-z)/2,用 C 语言定义输入变量，实现输出显示；用汇编语言实现多项式算法。

（2）源代码

lab.c

```
#include<stdio.h>
#include<stdlib.h>
Typedef unsigned int u32;
```

```
 Int main()
 {
  u32 x=2;
  u32 y=3;
  u32 z=1;
  printf(“%d\n”,polymath(x,y,z));
  return 0;
}
```

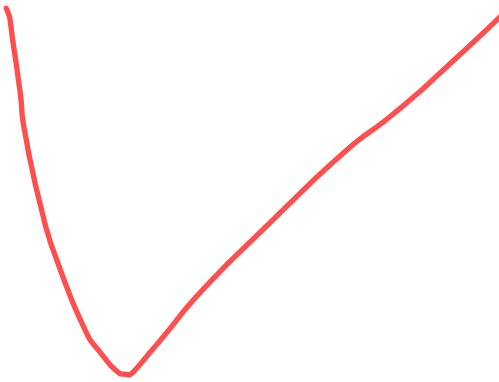
polymath.S

```
#include "polymath.h"
ENTRY(polymath)
add w0, w1, w0
lsl w0,w0,3
sub w0,w0,w2
lsr w0,w0,1
ret
ENDPROC(polymath)
```

（3）运行结果

x=2 y=3 z=1,输出结果为 19

```
Temporary breakpoint 2 at 0x400608: file lab3.c, line 7.
Starting program: /root/lab3

Temporary breakpoint 2, main () at lab3.c:7
7           u32 x=2;
(gdb) next
8           u32 y=3;
(gdb) next
9           u32 z=1;
(gdb) next
11          printf("%d\n",polymath(x,y,z));
(gdb) info locals
x = 2
y = 3
z = 1
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /root/lab3
19
[Inferior 1 (process 13252) exited normally]
(gdb)
```



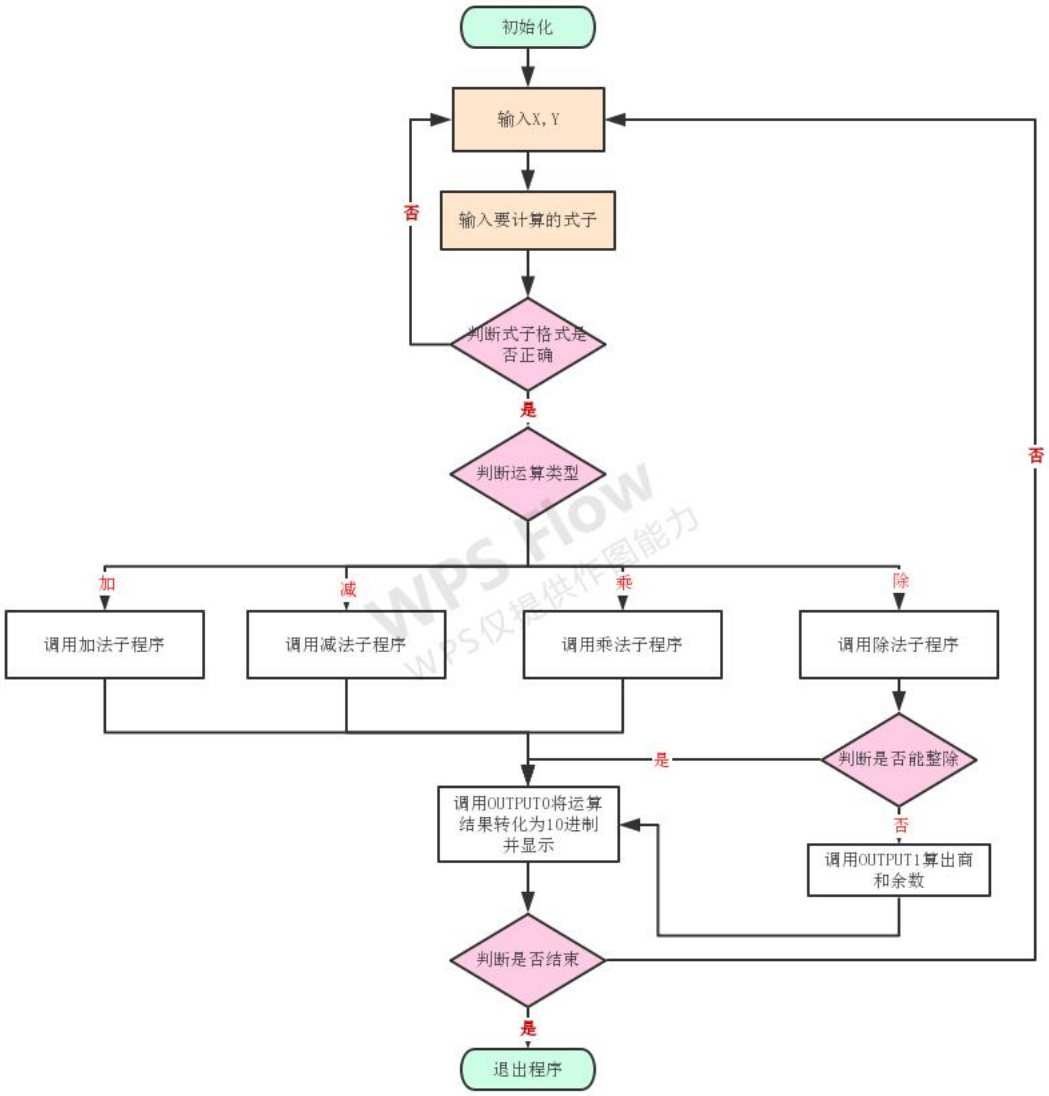
二、计算器实现

1. x86 实现

（1）设计思路

数据段定义变量和字符串，代码段进行多项式的计算。先定义 x, y，再分别对四种运算 a. 乘，b. 加，c. 减，d. 除，进行定义；对输入进行判定，判定其运算类型并一一对应，最终得出结果。

（2）程序框架



2.3

（3）源代码

```
DATA SEGMENT
X DB 0,0,0
Y DB 0,0,0
```

```
SIGNAL DB 0
STRING1 DB
'**************************************************
**************************',0DH,0AH,'$'
STRING2 DB 'X=',0DH,0AH,'$'
STRING3 DB 'Y=',0DH,0AH,'$'
STRING4 DB 'SIGNAL IS ',0DH,0AH,'$'
STRING5 DB 'RESULT=',0DH,0AH,'$'

DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
   MOV DS,AX
   XOR AX,AX
   XOR BX,BX
   XOR CX,CX
   XOR DX,DX

LEA DX,STRING2
MOV AH,09H
```

```
INT 21H
   LEA DI,X                ;取 X 地址
INPUT1:MOV AH,01H          ;把前向输入数字放在 X 对应的
数据段地址中
   INT 21H
   CMP AL,0DH
   JZ NEXT00
   INC BH
   MOV [DI],AL
   INC DI
   LOOP INPUT1
NEXT00:MOV [DI],0DH
   MOV [DI+1],0AH

MOV CL,04H                 ;显示输入 X 的值
MOV CH,00H
LEA DI,X
DISPLAY1:
   MOV DL,[DI]
   MOV AH,02H
   INT 21H
   ADD DI,1
```

```
LOOP DISPLAY1

LEA DX,STRING3
MOV AH,09H
INT 21H
   LEA DI,Y                    ;取Y地址
INPUT2:MOV AH,01H              ;把后向输入数字放在Y对应
的数据段地址中
   INT 21H
   CMP AL,0DH
   JZ NEXT01
   INC BH
   MOV [DI],AL
   INC DI
   LOOP INPUT2
NEXT01:MOV [DI],0DH
   MOV[DI+1],0AH

MOV CL,04H                     ;显示输入Y的值
MOV CH,00H
LEA DI,Y
DISPLAY2:
```

```
    MOV DL,[DI]
    MOV AH,02H
    INT 21H
    ADD DI,1
LOOP DISPLAY2


LEA DX,STRING4
MOV AH,09H
INT 21H
    LEA DI,SIGNAL          ;取符号地址
INPUT3:MOV AH,01H              ;输入符号
    INT 21H
    MOV [DI],AL
    ;MOV DL,AL              ;显示输入符号
    ;MOV AH,02H
    ;INT 21H
    MOV DL,0DH
    MOV AH,02H
    INT 21H
    MOV DL,0AH
    MOV AH,02H
```

```
    INT 21H

    MOV DL,[DI]
    CMP DL,'*'
    JZ NEXT1
    CMP DL,'+'
    JZ NEXT2
    CMP DL,'-'
    JZ NEXT3
    CMP DL,'/'
    JZ NEXT4

NEXT1:XOR AX,AX
    XOR BX,BX
    CALL ATOO                ;输入的 ASCII 转成两位数,
3536H--56,X=(AL),Y=(BL)
    MUL BL                   ;无符号乘法
    MOV DX,AX
    CALL OTOASC              ;计算结果转成 ASCII
JMP MAIN

NEXT2:XOR AX,AX
```

```
    XOR BX,BX
    CALL ATOO                    ;输入的ASCII转成两位数,
3536H--56,X=(AL),Y=(BL)
    ADD AL,BL
    JC CARRY
    JMP NEXT2O

CARRY:ADD AH,1

NEXT2O:MOV DX,AX        ;保护AX
    CALL OTOASC
JMP MAIN

NEXT3:XOR AX,AX
    XOR BX,BX
    XOR DX,DX
    CALL ATOO
    CMP AL,BL
    JB NEXT3O
    SUB AL,BL
    JMP NEXT31
NEXT3O: XCHG AL,BL
```

```
    SUB AL,BL
    NEG AL
    MOV AH,0FFH
NEXT31:MOV DX,AX    ;保护 AX
    CALL OTOASC
JMP MAIN
NEXT4:XOR AX,AX
    XOR BX,BX
    XOR DX,DX
    CALL ATOO
    DIV BL
    MOV DX,AX      ;保护 AX
    CALL OTOASC
JMP MAIN

MAIN:
LEA DX,STRING5
MOV AH,09H
INT 21H

XOR CX,CX
MOV CL,4
```

```
MOV DI,0013H                ;显示模块
LOOP1:
   MOV DL,[DI]
   MOV AH,02H
   INT 21H
   SUB DI,1
LOOP LOOP1
JMP END1

OTOASC PROC NEAR      ;十进制转成 ASCII 子程序
   PUSH AX
   PUSH DX
   MOV DX,AX        ;保护 AX
   AND AL,0FH        ;AL 低四位转成 ASCII
   CMP AL,'9'
   JG OTOA1
   ADD AL,30H
   JMP MAIN1
OTOA1:ADD AL,37H

MAIN1:XOR DI,DI    ;放入指定存储单元
   MOV DI,0010H
```

```
    MOV [DI],AL

    MOV AL,DL          ;AL 高四位转成 ASCII
    AND AL,0F0H
    SHR AL,4
    CMP AL,'9'
    JG OTOA2
    ADD AL,30H
    JMP MAIN2
OTOA2:ADD AL,37H

MAIN2:MOV [DI+1],AL      ;放入指定存储单元

    AND AH,0FH      ;AH 低四位转成 ASCII
    CMP AH,'9'
    JG OTOA3
    ADD AH,30H
    JMP MAIN3
OTOA3:ADD AH,37H

MAIN3:MOV [DI+2],AH
```

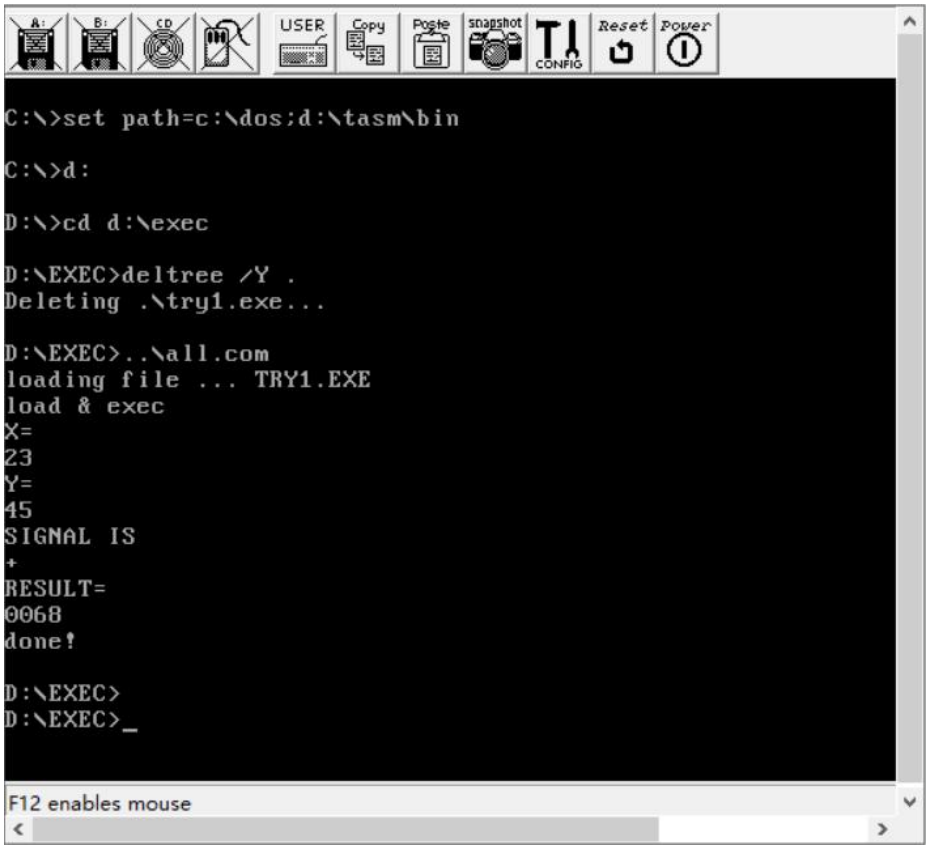
```
    MOV AH,DH      ;恢复 AH
    AND AH,0F0H     ;AH 高四位转成 ASCII
    SHR AH,4
    CMP AH,'9'
    JG OTOA4
    ADD AH,30H
    JMP MAIN4
OTOA4:ADD AH,37H

MAIN4:MOV [DI+3],AH

    POP DX
    POP AX
    RET
OTOASC ENDP

ATOO PROC NEAR                          ;输入的 ASCII 转成
两位数, 3536H--56,X=(AL),进位进到 AH,Y=(BL), 进位进到
BH
    PUSH CX
    XOR AX,AX
    LEA SI,X
```

```
    MOV AL,[SI]
    AND AL,0FH
    SHL AL,4
    MOV CL,[SI+1]
    AND CL,0FH
    ADD AL,CL
    JC ATOONEXT1
    JMP ATOONEXT2
ATOONEXT1: ADD AH,1

ATOONEXT2: XOR BX,BX
    LEA SI,Y
    MOV BL,[SI]
    AND BL,0FH
    SHL BL,4
    MOV CH,[SI+1]
    AND CH,0FH
    ADD BL,CH
    JC ATOONEXT3
    JMP ATOONEXT4
ATOONEXT3:  ADD BH,1
ATOONEXT4:
```

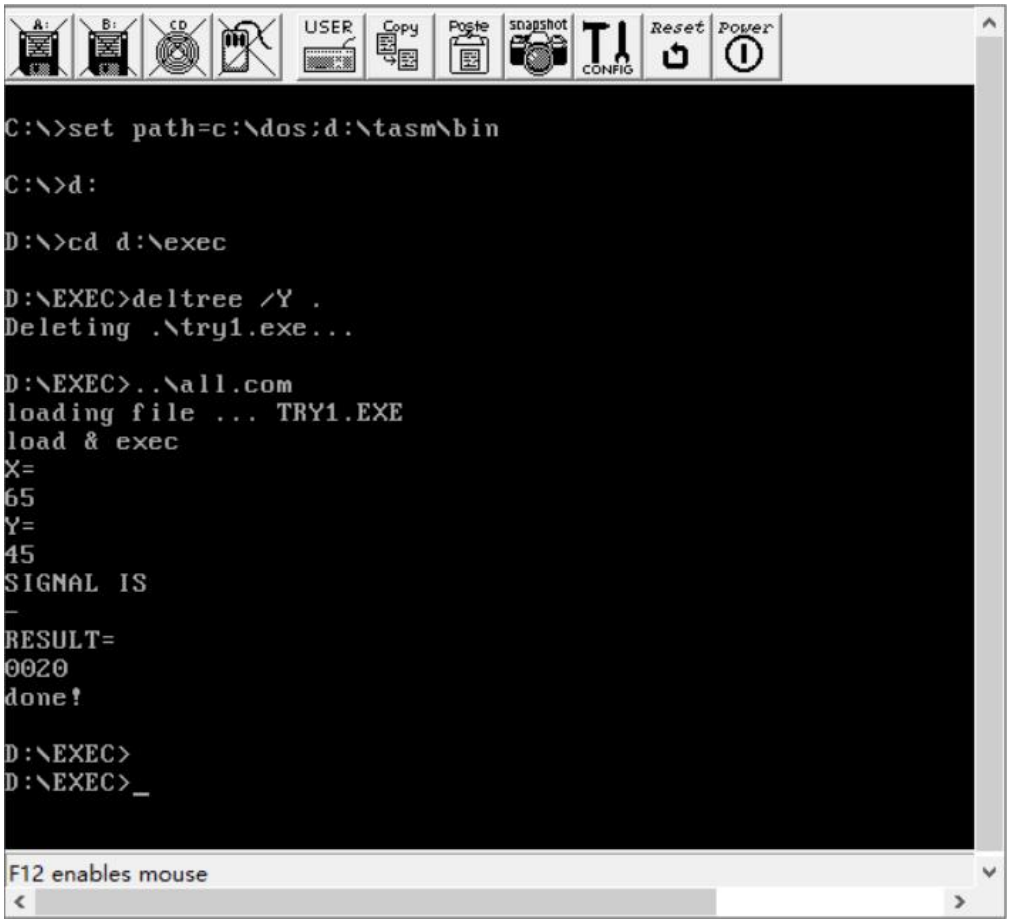
```
    POP CX
    RET
ATOO ENDP


END1:
    MOV AH,4CH
    INT 21H
CODE ENDS
END START
```

（4）运行结果

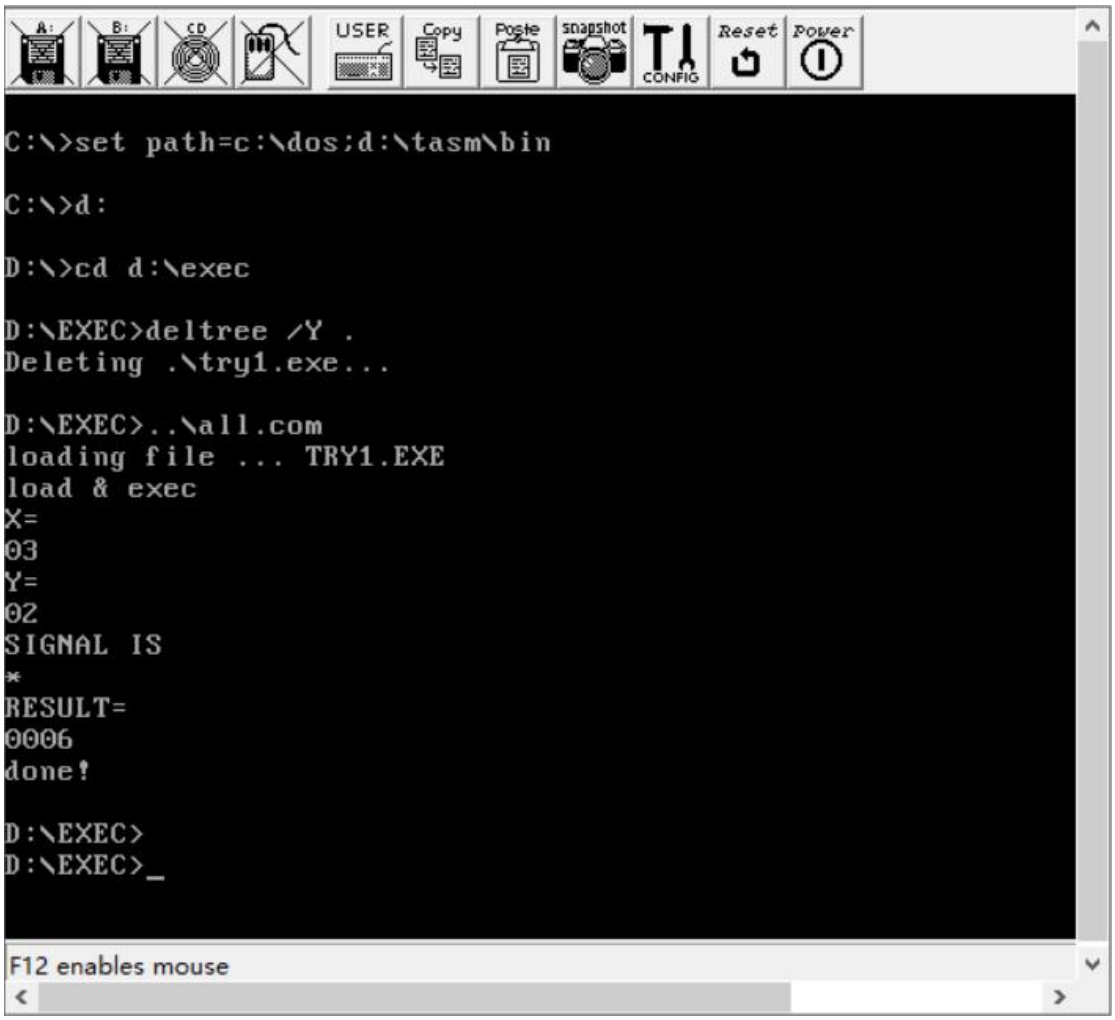
加　　x=23，y=45　　输出 68



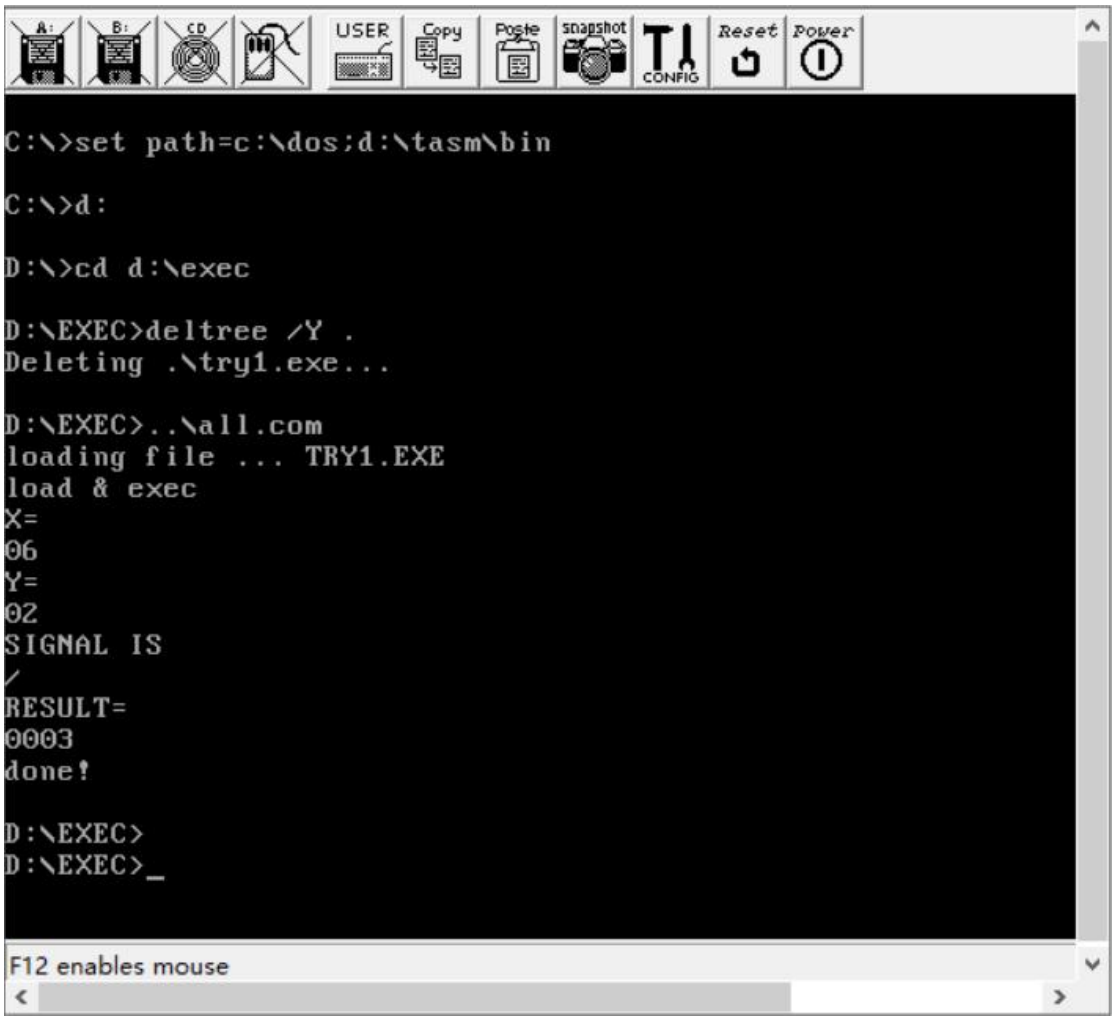
减　　x=65, y=45　　输出 20



乘　　x=3, y=2　　输出 6


A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
Power
C:\>set path=c:\dos;d:\tasm\bin
C:\>d:
D:\>cd d:\exec
D:\EXEC>deltree /Y .
Deleting .\try1.exe...
D:\EXEC>..\all.com
loading file ... TRY1.EXE
load & exec
X=
03
Y=
02
SIGNAL IS
*
RESULT=
0006
done!
D:\EXEC>
D:\EXEC>_
F12 enables mouse

除　　x=6, y=2　　输出 3

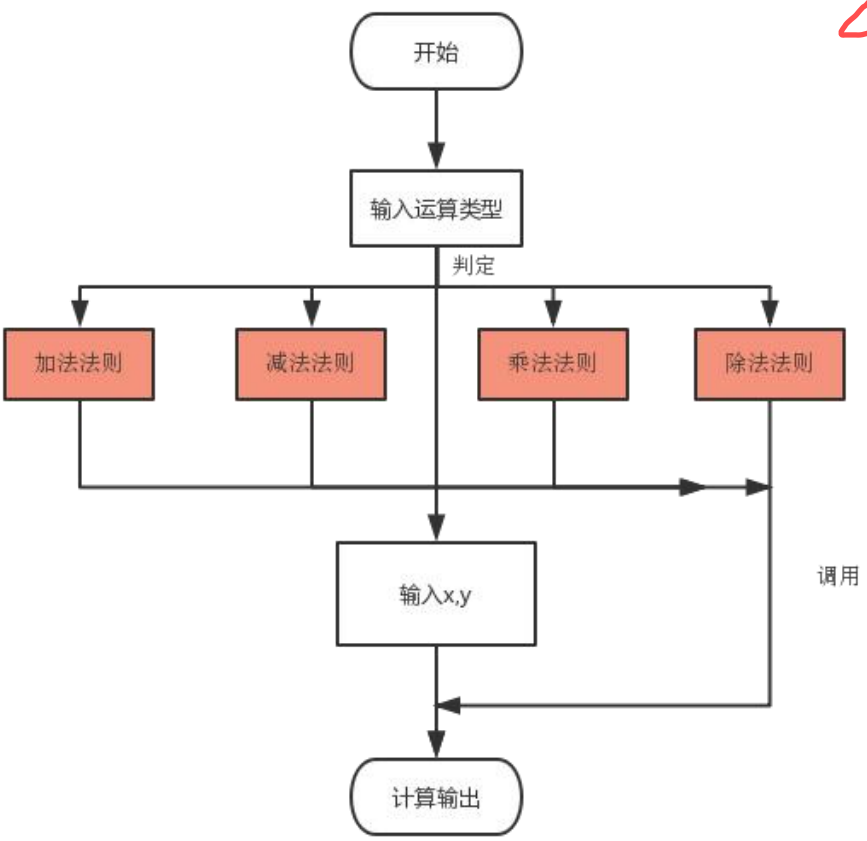

A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
Power
C:\>set path=c:\dos;d:\tasm\bin
C:\>d:
D:\>cd d:\exec
D:\EXEC>deltree /Y .
Deleting .\try1.exe...
D:\EXEC>..\all.com
loading file ... TRY1.EXE
load & exec
X=
06
Y=
02
SIGNAL IS
/
RESULT=
0003
done!
D:\EXEC>
D:\EXEC>_
F12 enables mouse

2. 华为云基于鲲鹏处理器 ARM V8 软件平台实现

（1）设计思路

用汇编对加减乘除四种运算法则进行定义，用 c 语言编写主函数，先选择运算类型，输入 x, y. 再调用运算法则, 对 x, y 进行相应的运算再输出。

（2）程序框图



（3）源代码

lab. c

```
#include <stdio.h>
#include <stdlib.h>
typedef unsigned int u32;
extern                int                add(u32,u32),
mul(u32,u32) ,divide(u32,u32), subtract(u32,u32);
int main()
{
    u32 x,y,a;
    while(1){
    printf("\nRule   of   the   function:\n   1.add\n
2.multiply\n 3.divide\n 4.subtract\n 5.quit\n");
    scanf("%d",&a);
    if(a==5){x=y=a=0; break;}
    else if(a==1){printf("add:\n");
    scanf("%d",&x);
    printf("%d + ",x);
    scanf("%d",&y);
    printf("%d + %d = %d",x,y,add(x,y));
    }
    else if(a==2){printf("multiply:\n");
    scanf("%d",&x);
    printf("%d * ",x);
```

```
    scanf("%d",&y);
    printf("%d * %d = %d",x,y,mul(x,y));
    }
    else if(a==3){printf("divide:\n");
    scanf("%d",&x);
    printf("%d / ",x);
    scanf("%d",&y);
    printf("%d / %d = %d",x,y,divide(x,y));
    }
    else if(a==4){printf("subtract:\n");
    scanf("%d",&x);
    printf("%d - ",x);
    scanf("%d",&y);
    printf("%d - %d = %d",x,y,subtract(x,y));
    }
   }
return 0;
}
```

polymath.h

```
#ifndef ENTRY
#define ENTRY(name) \
```

```
    .globl name ; \
    .align 4 ; \
    name:
#endif

/* If symbol 'name' is treated as a subroutine (gets
called, and returns)
 *  * then please use ENDPROC to mark 'name' as STT_FUNC
for the benefit of
 *    * static analysis tools such as stack depth
analyzer.
 *
 *   */
#ifndef ENDPROC
#define ENDPROC(name) \
    .type name, @function ; \
    .size name, .-name
#endif
```

add.S

```
#include "polymath.h"
ENTRY(add)
```

```
add w0, w1, w0
ret
ENDPROC(add)
```

divide.S

```
#include "polymath.h"
ENTRY(divide)
  mov w2,#0
A1:
  sub w0,w0,w1
  add w2,w2,#1
  cmp w0,#0
  bge A1
A2:
  sub w2,w2,#1
  mov w0,w2
ret
ENDPROC(divide)
```

mul.S

```
#include "polymath.h"
```

```
ENTRY(mul)
mul w0,w1,w0
ret
ENDPROC(mul)
```

subtract.S

```
#include "polymath.h"
ENTRY(subtract)
sub w0,w0,w1
ret
ENDPROC(subtract)
```

（4）运行结果

```
[root@595458152cb ~]# gcc -g lab.c add.S subtract.S mul.S divide.S -o lab
[root@595458152cb ~]# ./lab

Rule of the function:
 1.add
 2.multiply
 3.divide
 4.subtract
 5.quit
1
add:
12
12 + 36
12 + 36 = 48
Rule of the function:
 1.add
 2.multiply
 3.divide
 4.subtract
 5.quit
4
subtract:
48
48 - 22
48 - 22 = 26
Rule of the function:
 1.add
 2.multiply
 3.divide
 4.subtract
 5.quit
2
multiply:
26
26 * 32
26 * 32 = 832
Rule of the function:
 1.add
 2.multiply
 3.divide
 4.subtract
```