

微机原理实验五

PB13206106

罗勇冠

一.实验目的

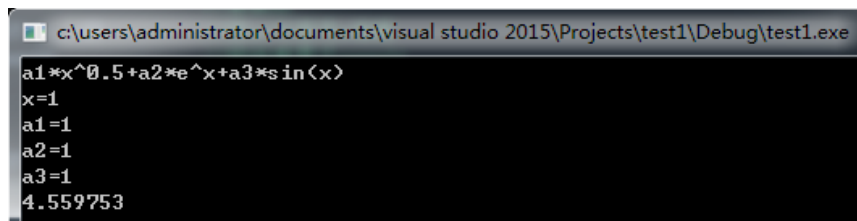
学会使用 32 位汇编进行编程,熟悉算术协处理器的数据传输指令、算术运算指令、比较指令、超越指令、常数指令和协处理器控制指令及其程序设计的方法。

二.实验内容

题 1: 从键盘输入 x 和 a1~a3, 计算 $a_1x^{1/2} + a_2e^x + a_3\sin(x)$, 结果输出到屏幕。当 x 小于 0 时, 请输出信息 “Error: x<0!”

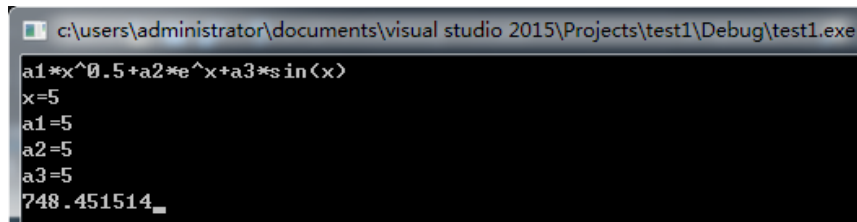
三.实验结果

测试 1: x=a1=a2=a3=1:



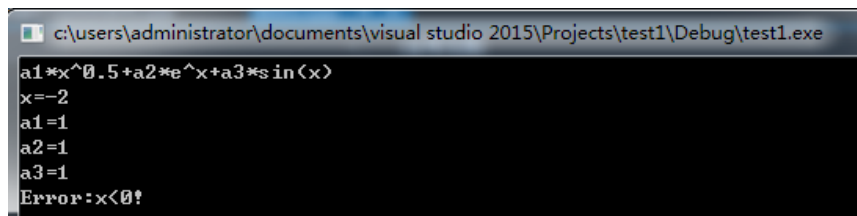
```
c:\users\administrator\documents\visual studio 2015\Projects\test1\Debug\test1.exe
a1*x^0.5+a2*e^x+a3*sin(x)
x=1
a1=1
a2=1
a3=1
4.559753
```

测试 2: x=a1=a2=a3=5:



```
c:\users\administrator\documents\visual studio 2015\Projects\test1\Debug\test1.exe
a1*x^0.5+a2*e^x+a3*sin(x)
x=5
a1=5
a2=5
a3=5
748.451514
```

测试 3: x=-2, a1=a2=a3=1:



```
c:\users\administrator\documents\visual studio 2015\Projects\test1\Debug\test1.exe
a1*x^0.5+a2*e^x+a3*sin(x)
x=-2
a1=1
a2=1
a3=1
Error:x<0!
```

四.实验分析

本次实验环境为 Visual Studio 2015, 使用了 32 位 C/C++ 嵌入汇编来编写包含算数协处理器运算指令的函数 FPU_arithmetic(), 传递参数为 a1, a2, a3 和 x, 返回值为 result。在该函数中, 首先判断 x 是否小于 0, 利用了 FTST 指令, 并判断比较结果中的状态寄存器是否为 C3=0, C2=0, C0=1, 是则表示 x 小于 0, 直接输出错误信息: Error: x<0! 然后结束程序。若 x 不是负数, 则按顺序计算 $a_1x^{1/2}$, a_2e^x , $a_3\sin(x)$ 三项, 再逐个将结果相加。

第一项 $a1 \cdot x^{1/2}$ 的计算主要使用了 FSQRT 指令得到 $x^{1/2}$ ，再用 FMULP 把 $a1$ 和 $x^{1/2}$ 相乘，结果存入 `result` 中。

第二项计算 $a2 \cdot e^x$ ，由于 FPU 中没有 e 的指数计算指令，故分步求 e^x ，先用 FLD2E 指令得到 $\log_2 E$ ，再用 FMULP 指令得到 $x \log_2 E$ ，将该结果拆分为整数部分和小数部分，取出控制寄存器的结果并把舍入控制 RC 置为 11，然后用指令 FRNDINT 得到舍入后的整数部分，用 FSUB 指令将原数减去整数部分得到小数部分，F2XM1 指令计算得到 $2^{(\text{小数部分})-1}$ ，再用 FLD1 和 FADD 得到 $2^{(\text{小数部分})}$ ，用 FSCALE 乘上 $2^{(\text{整数部分})}$ ，得到 e^x ，最后用 FMULP 指令乘上 $a2$ 得到 $a2 \cdot e^x$ 。

第三项计算 $a3 \cdot \sin(x)$ ，主要利用 FSIN 和 FMULP 指令得到结果。

至于输入和输出部分，直接利用 C/C++ 的 `scanf()` 和 `printf()` 函数来完成。

五. 意见建议

由于 emu8086 为 16 位汇编环境，无法实现算数协处理器运算指令，故采用了 C/C++ 的嵌入汇编方式来实现。在实现过程中，比较麻烦的是第二项 $a2 \cdot e^x$ 的计算，需要把计算步骤拆分为几个阶段来完成。但是通过本次上机加深了对算数协处理器程序设计的了解程度。

六. 源代码

```
#include "stdafx.h"
#include <stdio.h>
int error = 0;
double FPU_arithmetic(double a1, double a2, double a3, double x) {
    double result = 0;
    short temp;
    __asm {
        FLD x                //装载 x
        FTST                 //与 0 比较
        FSTSW AX              //取出状态寄存器
        AND AX, 4500H         //除 C3、C2、C0 外全清零
        CMP AX, 100H          //比较是否小于 0 (C3=0、C2=0、C0=1)
        JE UNDERZERO         //小于零则跳到报错
        FSQRT                 //把 x 开平方
        FLD a1                //装载 a1
        FMULP ST(1), ST       //把 a1 与根号 x 相乘
        FSTP result           //把结果置于 result

        FLD x
        FLDL2E                //装载 log2E
        FMULP ST(1), ST       //得到 xlog2E
        FST ST(1)             //复制结果
        FSTCW temp            //取出控制寄存器
        MOV     AX, temp
        PUSH AX               //保存原控制
        OR AX, 0C00H          //把舍入控制 RC 置为 11=截为 0
```

```

MOV temp, AX
FLDCW temp           //装载修改后的控制
FRNDINT              //舍入成整数
POP AX
MOV temp, AX
FLDCW temp           //还原控制
FXCH                 //将结果交换到栈顶
FSUB ST, ST(1)       //得到小数部分
F2XM1                //计算  $2^{(小数部分)-1}$ 
FLD1                 //加 1
FADD
FSCALE               //乘上  $2^{(整数部分)}$ 
FLD a2               //装载 a2
FMULP ST(1), ST      //得到  $a2e^x$ 
FLD result
FADD
FSTP result          //得到  $a1x^{0.5}+a2e^x$ 
FCOMP                //弹出

FLD x
FSIN                 //计算 x 的正弦
FLD a3               //装载 a3
FMULP ST(1), ST      //得到  $a3\sin x$ 
FLD result
FADD
FSTP result          //得到  $a1x^{0.5}+a2e^x+a3\sin x$ 

JMP OVER
UNDERZERO :
MOV error, 1
    OVER :
}
return result;
}

int main(void) {
    double a1, a2, a3, x, result;

    printf("a1*x^0.5+a2*e^x+a3*sin(x)\n");
    printf("x=");
    scanf("%lf", &x);
    printf("a1=");
    scanf("%lf", &a1);
    printf("a2=");

```

```
scanf("%lf", &a2);
printf("a3=");
scanf("%lf", &a3);

result = FPU_arithmetic(a1, a2, a3, x);
if (!error)
    printf("%lf", result);
else
    printf("Error:x<0!\n");
getchar();
getchar();
return 0;
}
```