

实验报告

课程名称: DSP 的原理与应用

实验名称: 实验 2.1 DSP 数据存取实验

专业-班级: 电气 1 班 学号: 220330124 姓名: 舒晟超

实验日期: 2024 年 5 月 7 日

试验台号: _____

报告总分数: _____

教师评语:

助教签字: _____

教师签字: _____

日 期: _____

一、实验目的

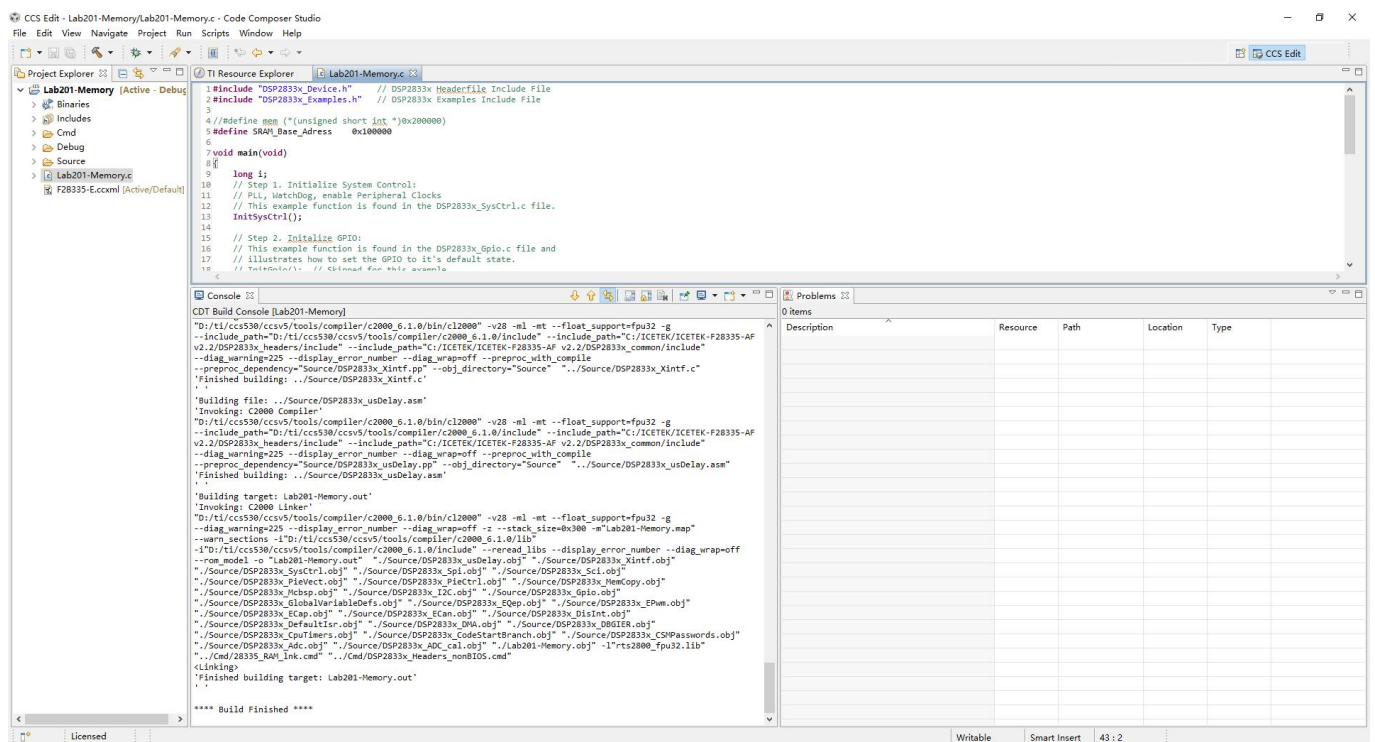
- 1.了解 TMS320F28335 的内部存储器空间的分配及指令寻址方式。
- 2.了解 ICETEK-F28335-AF 评估板扩展存储器空间寻址方法, 及其应用。
- 3.了解 ICETEK-F28335-AF 实验箱扩展存储器空间寻址方法, 及其应用。
- 4.学习用 Code Composer Studio 修改、填充 DSP 内存单元的方法。
- 5.学习操作 TMS32028xx 内存空间的指令。

二、实验过程

1. 导入工程，进行编译；
2. 打开反汇编窗口，查看函数入口程序；
3. 打开内存观察窗口，观察 SRAM 处的存储数据；
4. 添加断点，观察 SRAM 内的数据随程序运行的变化。

三、实验结果

1. 导入工程，进行编译；



- ## 2. 添加断点，调试程序：

- (1) main 函数入口地址为 FE02:

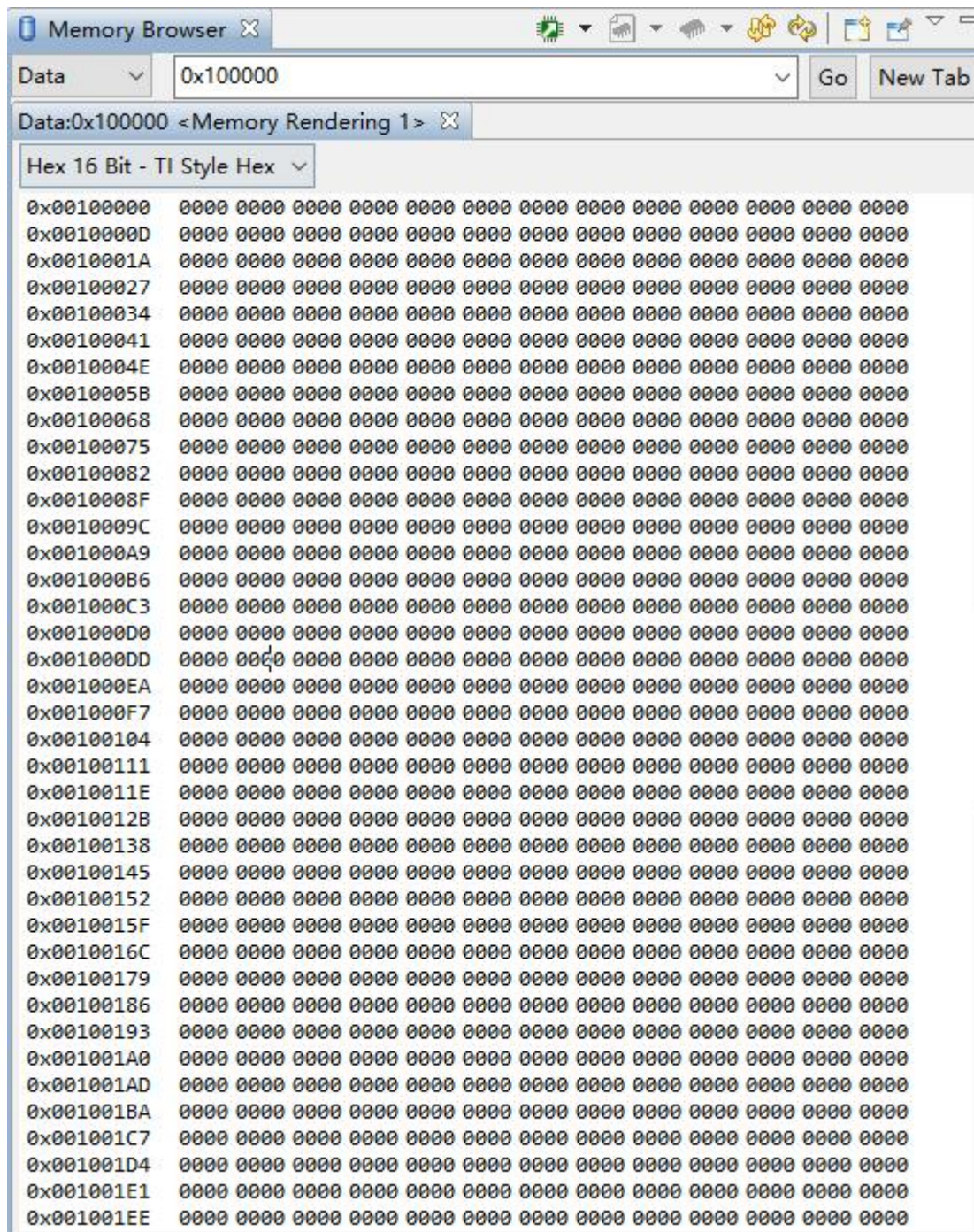
```

Disassembly
Enter location here

009e47: FF69 SPM #0
009e48: 0006 LRETR
70 PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
EnableInterrupts:
009e49: 761F0033 MOVW DP, #0x33
009e4b: 1A200001 OR @0x20, #0x0001
73 PieCtrlRegs.PIEACK.all = 0xFFFF;
009e4d: 2821FFFF MOV @0x21, #0xffff
76 EINT;
009e4f: 2910 CLRC INTM
78 }
009e50: FF69 SPM #0
009e51: 0006 LRETR
8 {
main:
009e52: FE02 ADDB SP, #2
13 InitSysCtrl();
009e53: 76409A8A LCR InitSysCtrl
19 InitXintf16Gpio(); //zq
009e55: 76409A30 LCR InitXintf16Gpio
23 DINT;
009e57: 3B10 SETC INTM
29 InitPieCtrl();
009e58: FF69 SPM #0
009e59: 76409E2A LCR InitPieCtrl
32 IER = 0x0000;
009e5b: 76260000 AND IER, #0x0000
33 IFR = 0x0000;
009e5d: 762F0000 AND IFR, #0x0000
36 for(i=0;i<=0xffff;i++)
009e5f: 0200 MOVB ACC, #0
009e60: 1E42 MOVL *-SP[2], ACC
009e61: 2901 CLRC SXM
009e62: FF20FFFF MOV ACC, #65535
009e64: 0F42 CMPL ACC, *-SP[2]
009e65: 6410 SB C$DW$L$_main$2$E, LT
38 *(int*)(SRAM_Base_Adress+i)=i;
C$DW$L$_main$2$B, C$L1:
009e66: 8F100000 MOVL XAR4, #0x100000
009e68: 8842 MOVZ AR6, *-SP[2]
009e69: A8A9 MOVL @ACC, XAR4
009e6a: 0742 ADDI ACC, *SP[2]

```

(2) XINTF 总线未使能前，SRAM 内数据如下图：



可见未初始化前，由于 SRAM 的掉电易失性，数据全为 0；

(3) InitXintfl6Gpio() 函数进行 XINTF 总线初始化后，SRAM 内数据如下：

Memory Browser	
Data	0x100000
Go	New Tab
Data:0x100000 <Memory Rendering 1>	
Hex 16 Bit - TI Style Hex	
0x00100000	1CCF A116 9FAC AE20 1906 DE80 39E4 2373 7404 2158 B03D 7A02 3E2A
0x0010000D	1E14 5C65 C23B 25C3 146E 9C29 F57F AC32 BA00 0664 828A 682F 2441
0x0010001A	A197 A463 4758 AC79 87F1 8017 25B0 7163 3A7B 1110 34D0 CA06 9A35
0x00100027	1F4E 05FA B91D D28D 66F1 5E56 112F 96C4 73C8 63D8 9E5F 44A0 C70A
0x00100034	6ED4 2288 7EC4 0E9A 777D 294C 3C25 9C86 B2CC AD7E 1DD0 102A 8907
0x00100041	6E22 3E35 AE7C 3B79 2612 A994 B080 0471 CB40 794F A74D EB1C 7178
0x0010004E	3C8C 3322 8B82 2148 00D4 07DE AF2D D8C9 8E20 B882 45E2 E30E 7869
0x0010005B	3474 4020 A8C6 2277 A2E4 1963 2198 1CCF EAEF F0EE 9E07 4808 E3D2
0x00100068	47A9 67A3 AA1B F2D6 29E9 2614 830B 20E8 4C2A F929 9378 42A8 A0A7
0x00100075	32CE B09C 6905 115D 21C0 EAC2 C230 9758 B439 4A7C C0EB 9862 03FF
0x00100082	8010 295B 03DF 8870 A646 6CC1 74A0 F4BC 0C02 8E76 AA68 7007 45A9
0x0010008F	A8D8 694D A902 C548 8160 D872 45B9 051A 644A FD6A C63C E3AE 729F
0x0010009C	2417 1289 1440 A36B BCD4 E66C 625E 2DE0 47CB FC24 C350 E401 51CE
0x001000A9	A2E3 7228 6589 6D01 2E5C F289 BB60 2CFA F88F 41DB 3010 DA9E 5860
0x001000B6	53C7 3A10 1C92 B902 54B0 E1B0 ACC3 C2DE 453C C4C5 1CA4 F883 6589
0x001000C3	00B1 6F88 6BE1 C861 180E AC53 48A8 CA56 AE33 192C 2150 6A1D 0407
0x001000D0	4187 1B3C 15EB 02D9 E292 0497 C7C6 EC42 663D 37A1 EA3F 3742 0832
0x001000DD	3CAA 7DB5 1E20 133A ABBF 3B19 DC16 8A9F DAF9 2968 12A3 8AA8 52C1
0x001000EA	44E0 C554 1241 62A8 0DED 5E20 0E92 BDBA DE0D 1FD4 0BFD 8A34 7169
0x001000F7	1009 3078 1B0F 72CB 52D3 2575 B102 CCDC 38C2 B338 F042 7C3D 0096
0x00100104	074B 386A BF33 C7E0 9BCB 886B 278F F004 489D 2230 5880 1BD8 8EDC
0x00100111	A513 C8CC C9D9 2818 6CFC 2649 9066 4D86 9C78 6E6A 3EA3 46B1 8524
0x0010011E	2888 228B 1375 8210 666E 9A1F 1274 12A9 4003 3852 4591 983A 3F12
0x0010012B	7A92 5A78 4118 BFCE F552 238C C707 03AF 8124 4F31 610B A660 E0AF
0x00100138	D17E 425D 114C 3DA8 2835 AD5F E072 964C 8FA0 E826 69D1 1E62 C5AE
0x00100145	82D8 7B2D 4783 8852 9609 819B 6103 A5F6 0244 8FDC DC0A 9EFE A297
0x00100152	6745 3C21 7968 28FF 769C EF00 FD60 A7D4 8C67 E0DD 4629 C2C2 9343
0x0010015F	9C27 00DD 3C9E 719A 0A81 6A9C 131C 0684 A7C6 88F0 0583 2920 6C35
0x0010016C	0667 002D E42B D40A 8CF8 8E97 2900 9CC6 4EB7 BD47 C147 C9D6 79E6
0x00100179	D346 9964 454F 5FC3 C464 0491 112C D2BE EC42 F86C 16B6 B260 994F
0x00100186	3465 39C7 637B 45CE 9978 3726 C770 2F60 14D9 07AC 08D2 71CB 8848
0x00100193	7065 5DF4 6E84 4960 BDE5 9E90 27A1 D82A 40B8 EC25 28D4 96C6 3445
0x001001A0	7262 B2B0 060E 57A1 4122 400D 2927 7311 BBD1 4B1A AF52 F228 D95D
0x001001AD	3594 A380 0458 CA36 1E80 B8D0 98AE C657 4D4E 7C64 98F4 50EC 9CCC
0x001001BA	BC95 FD59 5557 893E 90AF 4E02 11B5 36B6 E695 2ED6 D5D0 966D 520D
0x001001C7	00C8 7914 C08E 3273 570B 50D9 F943 8345 EF0B 2394 1446 2B68 894F
0x001001D4	D303 D34A 02A8 0A96 9E89 10F6 031F 80E0 CF8E 245F 042D 9413 268E
0x001001E1	8C32 7E81 1448 5A5A 6C4C EA92 CAC6 3608 045B 9EE6 801E 8325 63C3
0x001001EE	5815 2D49 6E81 D247 878F 173B 288B E002 3159 B8C0 8539 8AD2 7173

SRAM 内被填充随机数据；

- (4) 使用 Fill Memory 功能向 SRAM 内写入数据：

Fill Memory

Start Address: 0x100000

Memory Page: Data

Length: 0x0010

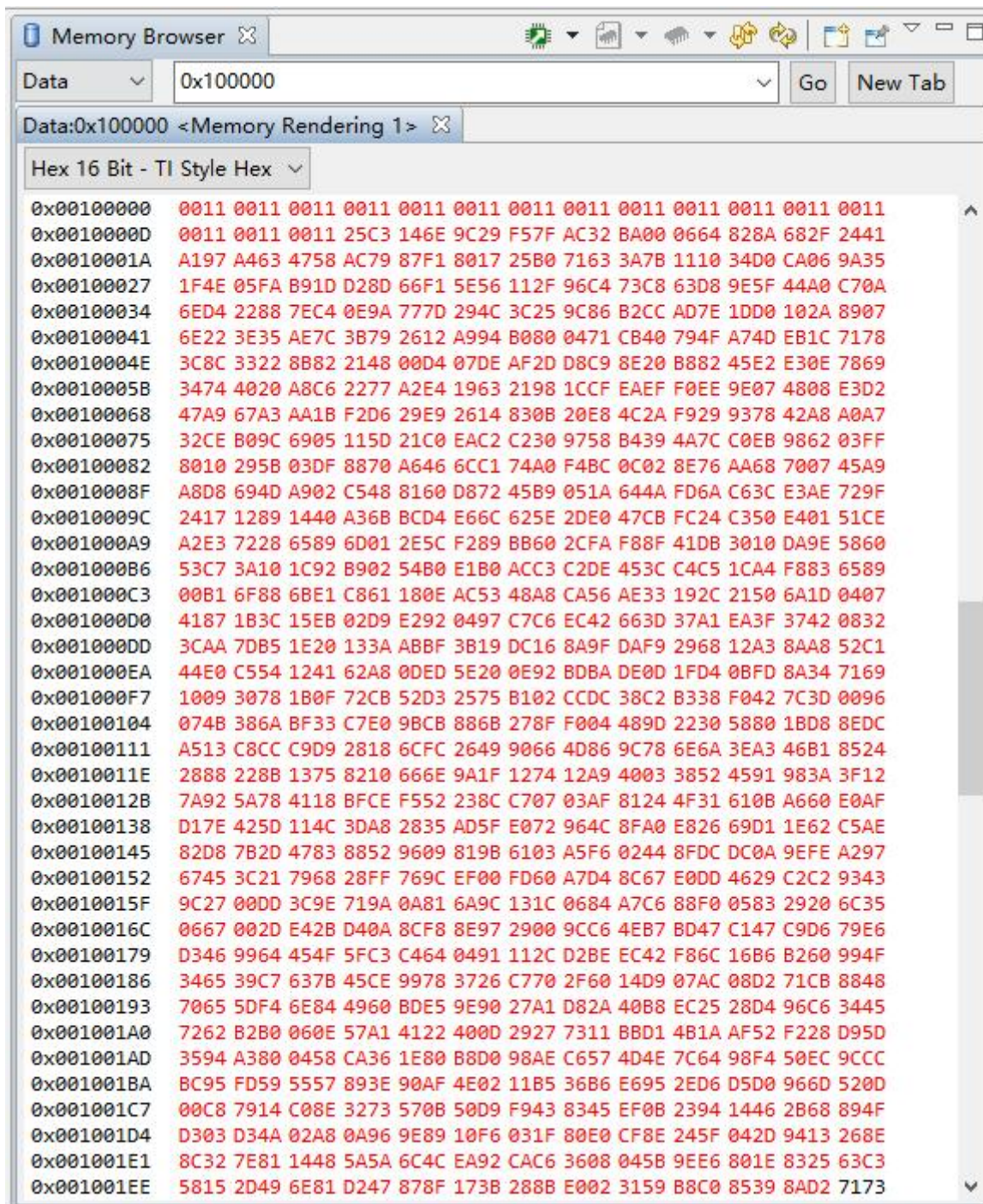
Data Value: 0x0011

Type-size: Default word size

?

OK Cancel

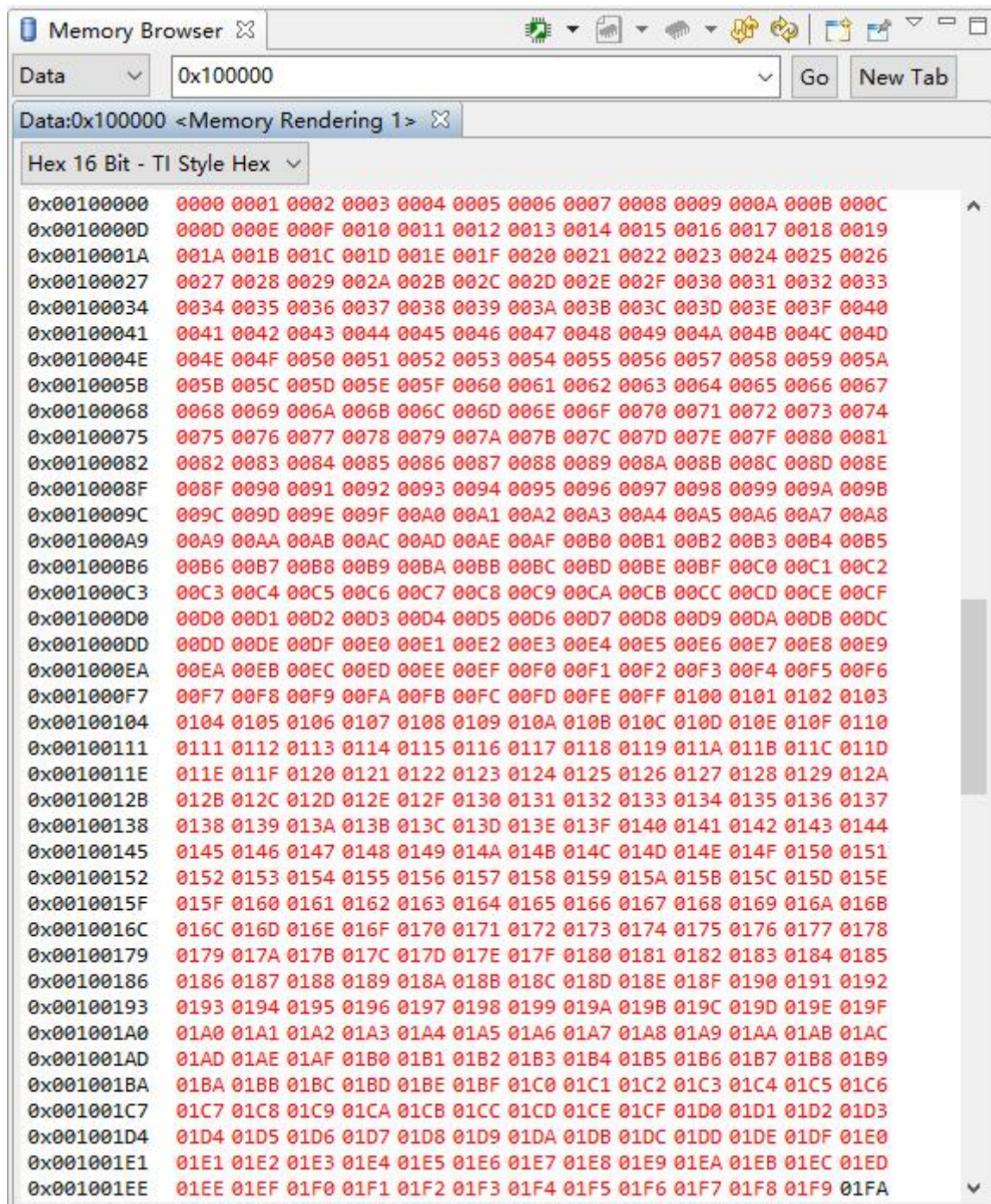
从 0x100000 开始写入，写入 16 个字节，写入值为 0x0011，结果如下：



(5) 使用程序向 SRAM 内写入数据:

```
// Step 4. Write Data to SRAM:
for(i=0;i<=0xffff;i++)
{
    *(int *)(SRAM_Base_Adress+i)=i;
};
```

上述代码的含义为: 向 SRAM 内写入数据, 从 SRAM 的基地址 SRAM_Base_Adress 开始, SRAM 内的每个值被赋值为当前地址的偏移量, 结果如下:



四、问题与思考

1、实验程序运行之后，为什么是 65536 个单元的数值被赋值了？（0.5 分）

```
// Step 4. Write Data to SRAM:
for(i=0;i<=0xffff;i++)
{
    *(int*)(SRAM_Base_Adress+i)=i;
};
```

此段代码中，for 循环执行了 0xffff 次（即 65535 次），每次执行 for 循环，SRAM 内的对应地址的值（基地址 SRAM_Base_Adress 加上由 i 确定的偏移量）被赋值为 i，执行 65535 次后，65536 个单元的数值被赋值了。

实验报告

课程名称: _____ DSP 的原理与应用

实验名称: _____ 实验 2.2 浮点运算

专业-班级: _____ 电气 1 班 学号: _____ 220330124 姓名: _____ 舒晟超

实验日期: _____ 2024 年 5 月 7 日

试验台号: _____

报告总分数: _____

教师评语:

助教签字: _____

教师签字: _____

日 期: _____

一、实验目的

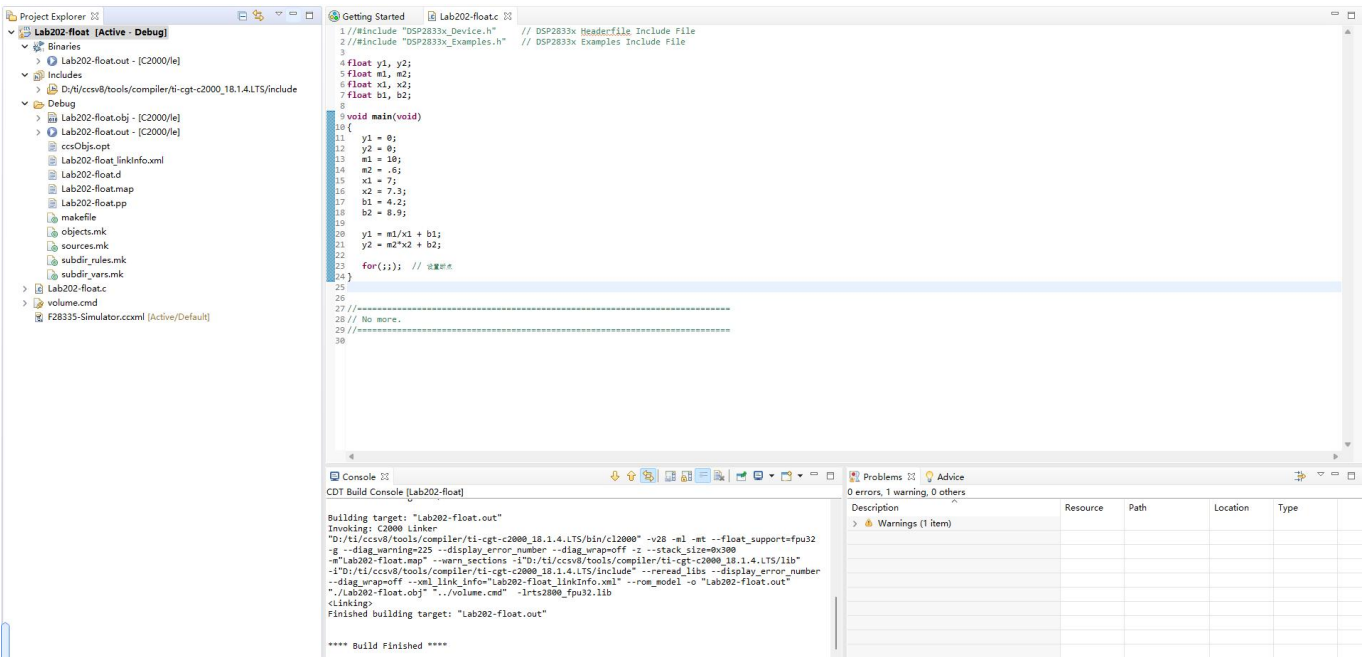
- 1. 了解 F28335 简单的浮点运算。
- 2. 熟悉浮点运算的编程。

二、实验过程

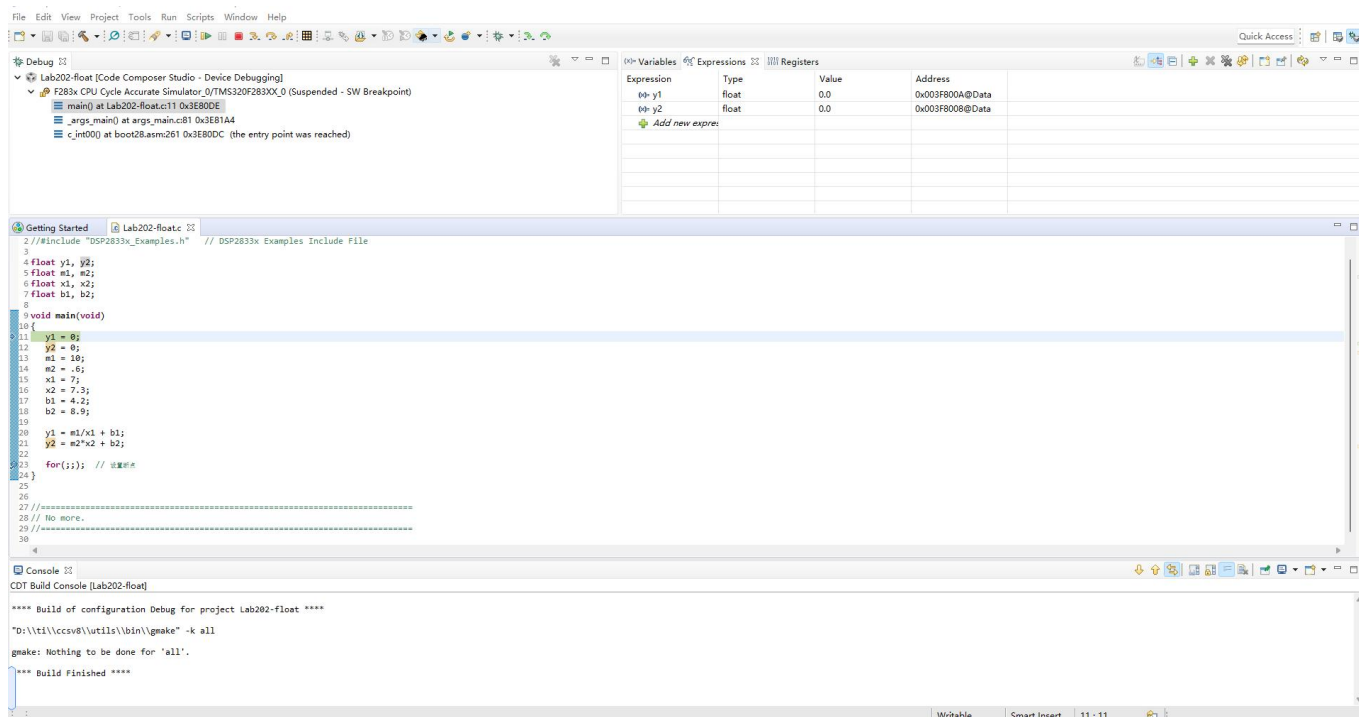
- 1. 配置 CCS 工程并加入软件仿真文件。
- 2. 阅读实验程序，编译并软件下载程序。
- 3. 设置断点，将 y1, y2 添加进入观察窗中。
- 4. 运行程序，观察 y1, y2 的变化。

三、实验结果

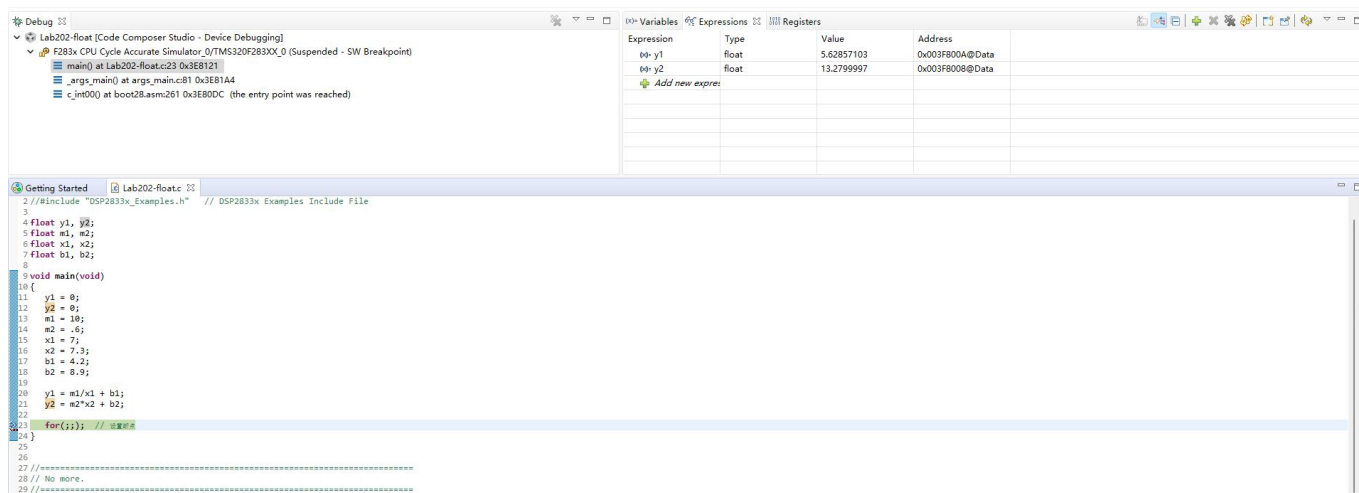
- 1. 编译程序，无报错。



- 2. 进行调试，添加断点和观察变量如图所示。



3. 进行浮点运算，结果如下图所示：



计算得到 $y_1 = 5.62857103$; $y_2 = 13.279997$.

四、问题与思考

1、做复杂的浮点运算。 $y = \sigma(w \cdot x + b)$ ，函数 σ 满足 $\sigma(x) = \begin{cases} x, & x \geq 0 \\ 0.01 * x, & x < 0 \end{cases}$ ，

已知 $w = [0.084674723, 0.81282741, 0.10292697, -0.047821075, -0.10049961, 1.1399955, 0.032412339, -0.039466377, 0.63574737, -0.099579401, 0.045932423, 0.056737024, -0.22110499, 0.61971980, -0.092473000, -0.32669711, 0.42802703, -0.21985172, 0.064885728, 1.2945132, -0.38374531, -0.094551362, 0.66813517, -0.42289069, -0.70333308, 0.10515465, 0.012392324, 0.17113267, 0.11922766, 1.0923374, -0.52938485, -0.26163799, 0.023534546, 1.3951927, -0.62430763, -0.34834740, 0.93761641, 0.99460375, -0.26750037, 0.40269896, 2.7415528, 0.16500330, 0.069107793, 0.37492210, -0.58213925, 0.17143276, 0.11286087, -0.28556311, -0.13933337, -0.025005803, 0.27282801, 0.85436279, -0.079539515, 0.048446644, 2.9071076, -0.043221515, -0.17358761, -0.71363419, -0.0023091321, 0.35966095, 0.22097406, -0.34137756, 0.094514966];$

b= [-0.43400329;6.0672159;-0.30059275;3.7516675;1.2958264;-2.5990894;3.3421099;-0.83062923;0.30215138];

x= [0.1;0.2;0.3;0.4;0.5;0.6;7];求 y 的值。（1 分）

题目要求进行矩阵运算并将运算后的矩阵用 Sigma 函数进行处理。

w 为 9 行 7 列矩阵； x 为 7 行 1 列矩阵； b 为 9 行 1 列矩阵。

```
1 float w[9][7] = {{0.084674723,0.81282741,0.10292697,-0.047821075,-0.10049961,1.1399955,0.032412339},
2                  {-0.039466377,0.63574737,-0.099579401,0.045932423,0.056737024,-0.22110499,0.61971980},
3                  {-0.092473000,-0.32669711,0.42802703,-0.21985172,0.064885728,1.2945132,-0.38374531},
4                  {-0.094551362,0.66813517,-0.42289069,-0.70333308,0.10515465,0.012392324,0.17113267},
5                  {0.11922766,1.0923374,-0.52938485,-0.26163799,0.023534546,1.3951927,-0.62430763},
6                  {-0.34834740,0.93761641,0.99460375,-0.26750037,0.40269896,2.7415528,0.16500330},
7                  {0.069107793,0.37492210,-0.58213925,0.17143276,0.11286087,-0.28556311,-0.13933337},
8                  {-0.025005803,0.27282801,0.85436279,-0.079539515,0.048446644,2.9071076,-0.043221515},
9                  {-0.17358761,-0.71363419,-0.0023091321,0.35966095,0.22097406,-0.34137756,0.094514966}};
10
11 float x[7][1] = {{0.1},
12                  {0.2},
13                  {0.3},
14                  {0.4},
15                  {0.5},
16                  {0.6},
17                  {7}};
18
19 float b[9][1] = {{-0.43400329},
20                  {6.0672159},
21                  {-0.30059275},
22                  {3.7516675},
23                  {1.2958264},
24                  {-2.5990894},
25                  {3.3421099},
26                  {-0.83062923},
27                  {0.30215138}};
```

定义 sigma 函数

```
31 float sigma_x(float x)
32 {
33     if(x >= 0.0f)
34     {
35         return x;
36     }
37     else
38     {
39         return (0.01 * x);
40     }
41 }
```

定义完成后在主程序中进行矩阵运算和函数处理，


```

43 /**
44  * main.c
45  */
46 int main(void)
47 {
48     float tmp[9][1] = {0};
49     // Caculate bx+w
50     int i,j,z;
51     for(i = 0;i < 9;i++)
52     {
53         for(j = 0;j < 1;j++)
54         {
55             tmp[i][j]=0;
56             for(z = 0; z < 7; z++)
57             {
58                 tmp[i][j] += w[i][z] * x[z][j];
59             }
60         }
61     }
62     for(i = 0;i < 9;i++)
63     {
64         tmp[i][0] += b[i][0];
65     }
66     // Sigma x
67     for(i = 0;i < 9;i++)
68     {
69         y[i][0] = sigma_x(tmp[i][0]);
70     }
71
72     for(;;)
73     {
74         ;
75     }
76 }
77

```

主程序如下

1. float w[9][7] = {{0.084674723,0.81282741,0.10292697,-0.047821075,-0.10049961,1.1399955,0.032412339},
2. {-0.039466377,0.63574737,-0.099579401,0.045932423,0.056737024,-0.22110499,0.61971980},
3. {-0.092473000,-0.32669711,0.42802703,-0.21985172,0.064885728,1.2945132,-0.38374531},
4. {-0.094551362,0.66813517,-0.42289069,-0.70333308,0.10515465,0.012392324,0.17113267},
5. {0.11922766,1.0923374,-0.52938485,-0.26163799,0.023534546,1.3951927,-0.62430763},
6. {-0.34834740,0.93761641,0.99460375,-0.26750037,0.40269896,2.7415528,0.16500330},
7. {0.069107793,0.37492210,-0.58213925,0.17143276,0.11286087,-0.28556311,-0.13933337},
8. {-0.025005803,0.27282801,0.85436279,-0.079539515,0.048446644,2.9071076,-0.043221515},
9. {-0.17358761,-0.71363419,-0.0023091321,0.35966095,0.22097406,-0.34137756,0.094514966}};
- 10.
11. float x[7][1] = {{0.1},
12. {0.2},

```

13.         {0.3},
14.         {0.4},
15.         {0.5},
16.         {0.6},
17.         {7}};
18.
19. float b[9][1] = {{-0.43400329},
20.                 {6.0672159},
21.                 {-0.30059275},
22.                 {3.7516675},
23.                 {1.2958264},
24.                 {-2.5990894},
25.                 {3.3421099},
26.                 {-0.83062923},
27.                 {0.30215138}};
28.
29. float y[9][1];
30.
31. float sigma_x(float x)
32. {
33.     if(x >= 0.0f)
34.     {
35.         return x;
36.     }
37.     else
38.     {
39.         return (0.01 * x);
40.     }
41. }
42.
43. /**
44.  * main.c
45.  */
46. int main(void)
47. {
48.     float tmp[9][1] = {0};
49.     // Caculate bx+w
50.     int i,j,z;
51.     for(i = 0;i < 9;i++)
52.     {
53.         for(j = 0;j < 1;j++)
54.         {
55.             tmp[i][j]=0;
56.             for(z = 0; z < 7; z++)
57.             {
58.                 tmp[i][j] += w[i][z] * x[z][j];
59.             }
60.         }

```



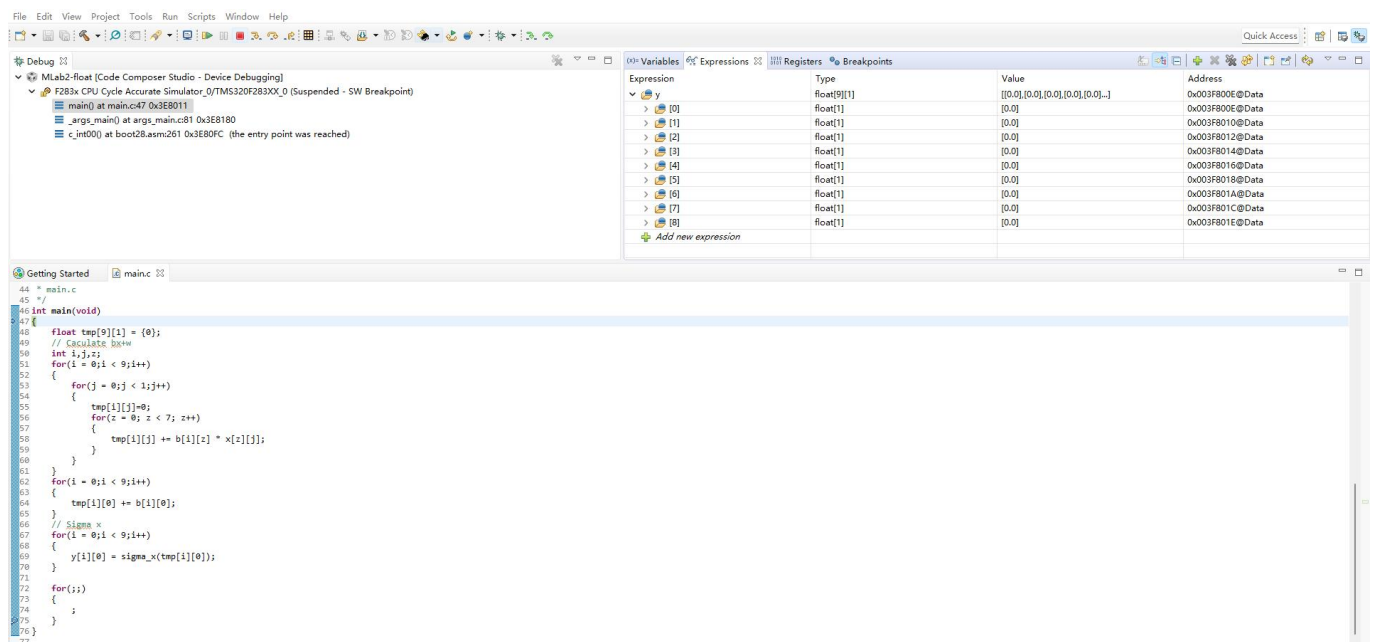
```

61.     }
62.     for(i = 0;i < 9;i++)
63.     {
64.         tmp[i][0] += b[i][0];
65.     }
66.     // Sigma x
67.     for(i = 0;i < 9;i++)
68.     {
69.         y[i][0] = sigma_x(tmp[i][0]);
70.     }
71.
72.     for(;;)
73.     {
74.         ;
75.     }
76. }

```

编译通过无报错。

进行调试，加入断点和监视变量如图所示：



运行代码至断点处。得到计算结果如图所示。

[illegible]