FIB.asm

执行FIB.asm, 会输出如下所示的信息：

38@00003000: $16 <= 00000000

42@00003004: $16 <= 00000080

46@0000300c: \*00000000 <= 00000000

46@00003008: $ 4 <= 00000000

54@00003010: $17 <= 00000000

58@00003014: $17 <= 00000001

62@00003018: $18 <= 00000000

66@0000301c: $18 <= 00000004

70@00003024: \*00000004 <= 00000001

70@00003020: $ 4 <= 00000004

78@00003028: $ 8 <= 00000000

82@0000302c: $ 8 <= 00000008

98@00003038: $ 9 <= 00000004

102@0000303c: $10 <= 00000000

106@00003040: $11 <= 00000001

110@00003044: $12 <= 00000000

118@0000304c: \*00000008 <= 00000001

118@00003048: $13 <= 00000001

126@00003050: $ 8 <= 0000000c

146@00003038: $ 9 <= 00000008

150@0000303c: $10 <= 00000004

154@00003040: $11 <= 00000001

158@00003044: $12 <= 00000001

166@0000304c: \*0000000c <= 00000002

166@00003048: $13 <= 00000002

174@00003050: $ 8 <= 00000010

194@00003038: $ 9 <= 0000000c

198@0000303c: $10 <= 00000008

202@00003040: $11 <= 00000002

206@00003044: $12 <= 00000001

214@0000304c: \*00000010 <= 00000003

214@00003048: $13 <= 00000003

222@00003050: $ 8 <= 00000014

242@00003038: $ 9 <= 00000010

246@0000303c: $10 <= 0000000c

250@00003040: $11 <= 00000003

254@00003044: $12 <= 00000002

262@0000304c: \*00000014 <= 00000005

262@00003048: $13 <= 00000005

270@00003050: $ 8 <= 00000018

290@00003038: $ 9 <= 00000014

294@0000303c: $10 <= 00000010

298@00003040: $11 <= 00000005

302@00003044: $12 <= 00000003

310@0000304c: \*00000018 <= 00000008

310@00003048: $13 <= 00000008

318@00003050: $ 8 <= 0000001c

338@00003038: $ 9 <= 00000018

342@0000303c: $10 <= 00000014

346@00003040: $11 <= 00000008

350@00003044: $12 <= 00000005

358@0000304c: \*0000001c <= 0000000d

358@00003048: $13 <= 0000000d

366@00003050: $ 8 <= 00000020

386@00003038: $ 9 <= 0000001c

390@0000303c: $10 <= 00000018

394@00003040: $11 <= 0000000d

398@00003044: $12 <= 00000008

406@0000304c: \*00000020 <= 00000015

406@00003048: $13 <= 00000015

414@00003050: $ 8 <= 00000024

434@00003038: $ 9 <= 00000020

438@0000303c: $10 <= 0000001c

442@00003040: $11 <= 00000015

446@00003044: $12 <= 0000000d

454@0000304c: \*00000024 <= 00000022

454@00003048: $13 <= 00000022

462@00003050: $ 8 <= 00000028

482@00003038: $ 9 <= 00000024

486@0000303c: $10 <= 00000020

490@00003040: $11 <= 00000022

494@00003044: $12 <= 00000015

502@0000304c: \*00000028 <= 00000037

502@00003048: $13 <= 00000037

510@00003050: $ 8 <= 0000002c

530@00003038: $ 9 <= 00000028

534@0000303c: $10 <= 00000024

538@00003040: $11 <= 00000037

542@00003044: $12 <= 00000022

550@0000304c: \*0000002c <= 00000059

550@00003048: $13 <= 00000059

558@00003050: $ 8 <= 00000030

578@00003038: $ 9 <= 0000002c

582@0000303c: $10 <= 00000028

586@00003040: $11 <= 00000059

590@00003044: $12 <= 00000037

598@0000304c: \*00000030 <= 00000090

598@00003048: $13 <= 00000090

606@00003050: $ 8 <= 00000034

626@00003038: $ 9 <= 00000030

630@0000303c: $10 <= 0000002c

634@00003040: $11 <= 00000090

638@00003044: $12 <= 00000059

646@0000304c: \*00000034 <= 000000e9

646@00003048: $13 <= 000000e9

654@00003050: $ 8 <= 00000038

674@00003038: $ 9 <= 00000034

678@0000303c: $10 <= 00000030

682@00003040: $11 <= 000000e9

686@00003044: $12 <= 00000090

694@0000304c: \*00000038 <= 00000179

694@00003048: $13 <= 00000179

702@00003050: $ 8 <= 0000003c

722@00003038: $ 9 <= 00000038

726@0000303c: $10 <= 00000034

730@00003040: $11 <= 00000179

734@00003044: $12 <= 000000e9

742@0000304c: \*0000003c <= 00000262

742@00003048: $13 <= 00000262

750@00003050: $ 8 <= 00000040

770@00003038: $ 9 <= 0000003c

774@0000303c: $10 <= 00000038

778@00003040: $11 <= 00000262

782@00003044: $12 <= 00000179

790@0000304c: \*00000040 <= 000003db

790@00003048: $13 <= 000003db

798@00003050: $ 8 <= 00000044

818@00003038: $ 9 <= 00000040

822@0000303c: $10 <= 0000003c

826@00003040: $11 <= 000003db

830@00003044: $12 <= 00000262

838@0000304c: \*00000044 <= 0000063d

838@00003048: $13 <= 0000063d

846@00003050: $ 8 <= 00000048

866@00003038: $ 9 <= 00000044

870@0000303c: $10 <= 00000040

874@00003040: $11 <= 0000063d

878@00003044: $12 <= 000003db

886@0000304c: \*00000048 <= 00000a18

886@00003048: $13 <= 00000a18

894@00003050: $ 8 <= 0000004c

914@00003038: $ 9 <= 00000048

918@0000303c: $10 <= 00000044

922@00003040: $11 <= 00000a18

926@00003044: $12 <= 0000063d

934@0000304c: \*0000004c <= 00001055

934@00003048: $13 <= 00001055

942@00003050: $ 8 <= 00000050

962@00003038: $ 9 <= 0000004c

966@0000303c: $10 <= 00000048

970@00003040: $11 <= 00001055

974@00003044: $12 <= 00000a18

982@0000304c: \*00000050 <= 00001a6d

982@00003048: $13 <= 00001a6d

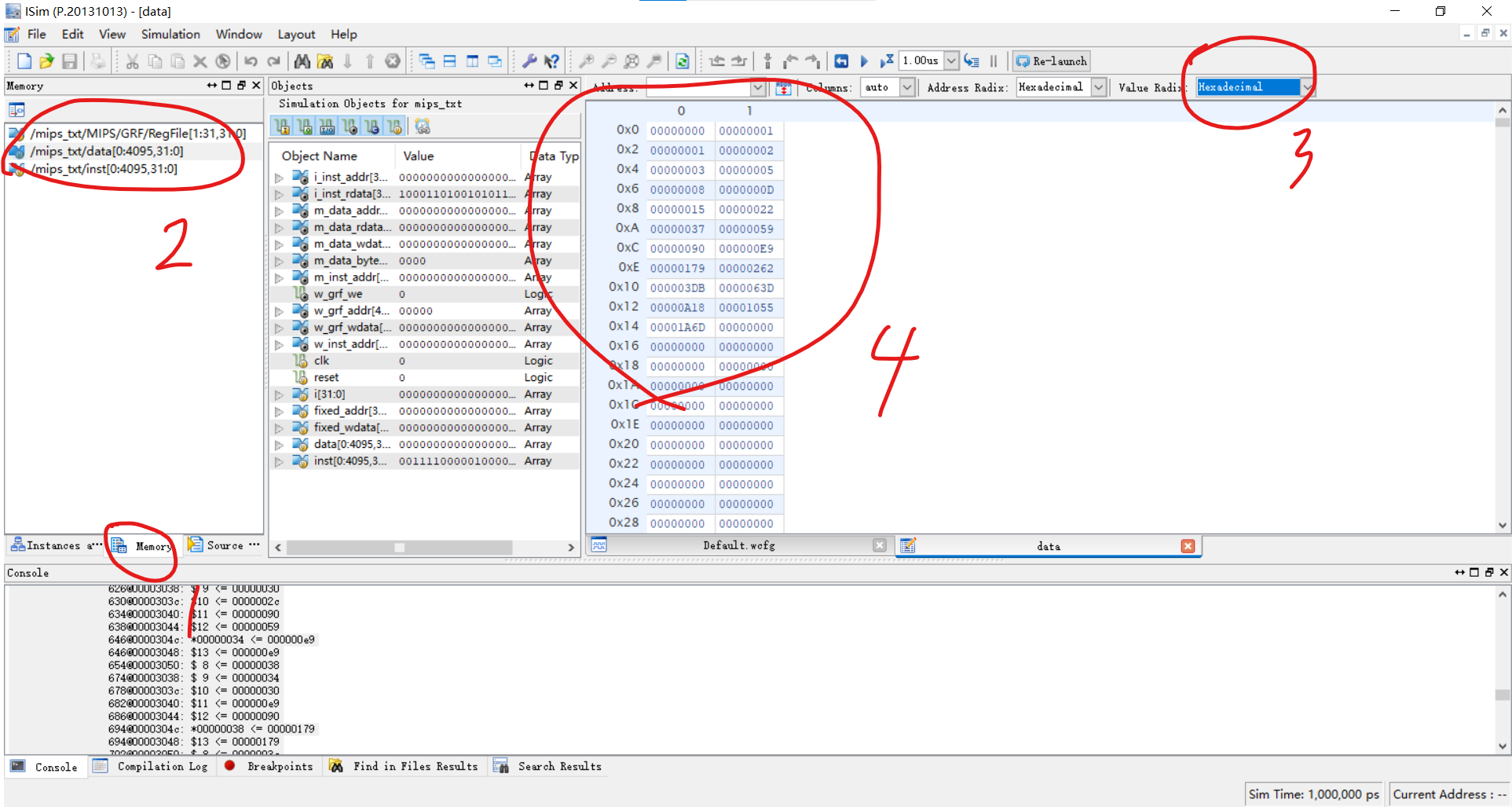
990@00003050: $ 8 <= 00000054

主要是看输出的内存写入信息（标红色部分）是否是一个斐波那契数列。

内存写入地址从0开始，依次递增4。

写入的数据构成斐波那契数列，依次是：（16进制）

1. 1，1，2，3，5,8，d,15,22,37,59,90,e9,179,…(后面的可以自行验证)

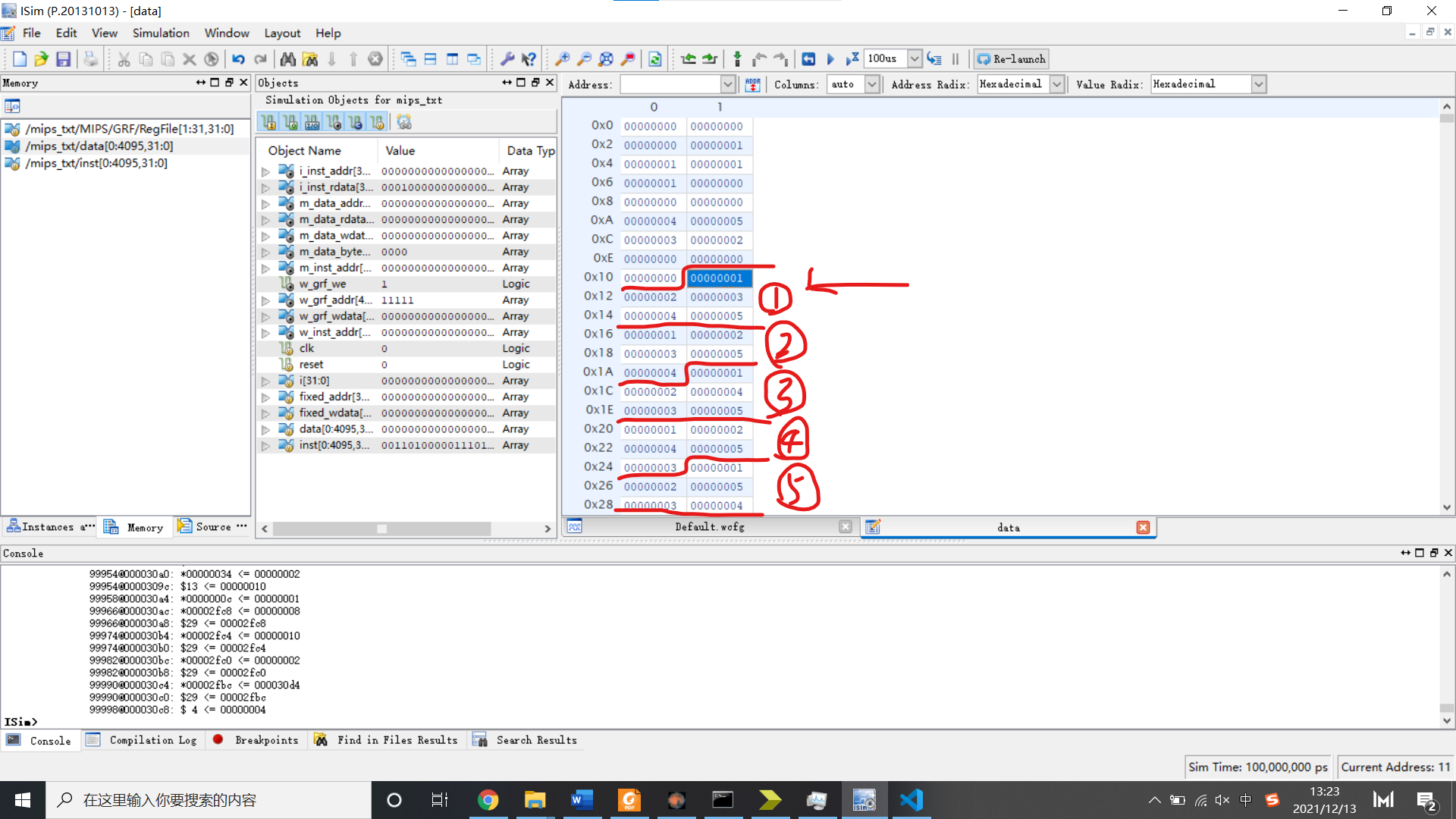


或者根据上图所示的方法直接查看数据存储器的值。

full.asm

修改全排列阶数的方法：改代码第15行的$s2数值，改成几就计算几的全排列。

默认是5，即计算5的全排列。



执行代码，要显示完整的全排列计算结果，模拟的时间应该要长，推荐把时间设置为100us以上。

因为不能像MARS那样直接在控制台输出，我选择在内存里存储全排列的结果，存储的起始位置如图所示，大概是0x11地址。

图中代码运行时设置的全排列阶数是5，则打印结果的时候5个为一组开始打印，如上图所示。依次是12345,12354,12435,12453,12534，… 若显示不全（即不到54321），可以尝试增加模拟的时间，在**右上角可以调节，Re-launch左面。**