1 Phase 4

1.1 条件 1：数量限制

<phase_ 4 > 函数中调用了 <read _ six_ numbers>, 不妨先从该被调用函数出发进行分析。

本文先选取 <read _ six_ numbers > 函数汇编代码的部分重点片段进行分析, 内容如下:

```
10548:   21f000ef            jal ra,10f66<sscanf>
1054c:   87aa                mv  a5,a0
1054e:   fef42623            sw  a5,-20(s0)
10552:   fec42783            lw  a5,-20(s0)
10556:   0007871b            sext.w  a4,a5
1055a:   4799                li  a5,6
1055c:   00f70463            beq a4,a5,10564
```

汇编代码中嵌套调用了函数 <sscanf >, 该函数前面已有涉猎。若设 Phase4 中输入数的个数为 NumOfNums，则该函数的运行结果是将 NumOfNums 的值保存到寄存器 a0 中。

现已知 <sscanf > 函数的返回值，将该函数及其后的汇编代码转换为伪代码:

```
a0=sscanf()  #即a0=NumOfNums
a5=a0
memory[s0-20]=a5
a5=memory[s0-20]
a4=a5
a5=6
if(a4==a5){
    end
}
else{
    explode_bomb()
}
```

在进行 if 判断语句前，a4==NumOfNums,a5==6, 所以要使 <explode_bomb > 函数不被调用, 则输入数的个数应等于 6，得到 Phase4 输入应满足的第一个条件。

1.2 条件 2：关系限制

回到 <phase_ 4 > 函数中，本文从 <read _ six_ numbers> 函数调用位置继续向下分析。选取关键部分汇编代码，并根据跳转语句及其跳转目的地的位置将选取的汇编代码分成 4 部分。

拆分后代码如下:

```
1041c:   0e0000ef            jal ra,104fc <read_six_numbers>
# part(1)
10420:   fc040793            addi    a5,s0,-64
10424:   fef43423            sd  a5,-24(s0)
10428:   fc040793            addi    a5,s0,-64
1042c:   07d1                addi    a5,a5,20
1042e:   fef43023            sd  a5,-32(s0)
10432:   a805                j   10462 <phase_4+0x5c>

```

```
10      # part(2)
11      10434:   fe843783            ld  a5,-24(s0)
12      10438:   439c                lw  a5,0(a5)
13      1043a:   0017979b            slliw   a5,a5,0x1
14      1043e:   fcf42e23            sw  a5,-36(s0)
15      10442:   fe843783            ld  a5,-24(s0)
16      10446:   0791                addi    a5,a5,4
17      10448:   4398                lw  a4,0(a5)
18      1044a:   fdc42783            lw  a5,-36(s0)
19      1044e:   2781                sext.w  a5,a5
20      10450:   00e78463            beq a5,a4,10458 <phase_4+0x52>
21      10454:   dd1ff0ef            jal ra,10224 <explode_bomb>
22
23      # part (3)
24      10458:   fe843783            ld  a5,-24(s0)
25      1045c:   0791                addi    a5,a5,4
26      1045e:   fef43423            sd  a5,-24(s0)
27
28
29      # part (4)
30      10462:   fe843703            ld  a4,-24(s0)
31      10466:   fe043783            ld  a5,-32(s0)
32      1046a:   fcf765e3            bltu    a4,a5,10434 <phase_4+0x2e>
33
34      # other
```

假设输入符合 Phase4 要求，则过程中 <explode_bomb > 函数不会被调用，那么程序会按照如下顺序执行。

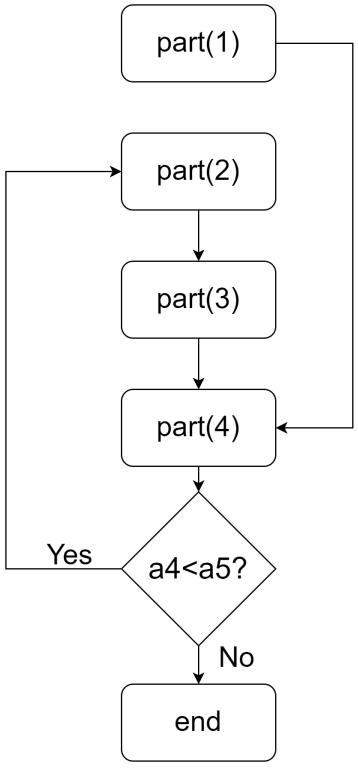

图 1: Phase4 汇编流程图

转化为伪代码：

```
a5=s0-64
Memory[s0-24]=a5
a5=s0-64
a5=a5+20
Memory[s0-32]=a5

a4=Memory[s0-24]
a5=Memory[s0-32]

while(a4<a5)
{
    a5=Memory[s0-24]
    a5=Memory[a5]
    a5=a5*2
    Memory[s0-36]=a5
    a5=Memory[s0-24]
    a5=a5+4
    a4=Memory[a5]
    a5=Memory[s0-36]
    if(a5!=a4){Bomb();}

    a5=Memory[s0-24]
    a5=a5+4
    Memory[s0-24]=a5

    a4=Memory[s0-24]
    a5=Memory[s0-32]
}

end
```

根据伪代码可知：

在第 1 次循环中，进行 if 判断前，$a5 = Memory[s0-64] \times 2$ ，$a4 = Memory[s0-60]$。if 判断中，要使炸弹不爆炸，应有 $a5 = a4$ 成立，即 $Memory[s0-64] \times 2 = Memory[s0-60]$。

在第 2 次循环中，进行 if 判断前，$a5 = Memory[s0-60] \times 2$ ，$a4 = Memory[s0-56]$。if 判断中，要使炸弹不爆炸，应有 $a5 = a4$ 成立，即 $Memory[s0-60] \times 2 = Memory[s0-56]$。

……

在第 n 次循环中，进行 if 判断前，$a5 = Memory[s0-68+4n] \times 2$ ，$a4 = Memory[s0-64+4n]$。if 判断中，要使炸弹不爆炸，应有 $a5 = a4$ 成立，即

$$Memory[s0-68+4n] \times 2 = Memory[s0-64+4n] \quad (1 \leq n \leq 5)$$

已知输入的 6 个数按顺序由内存地址低位向高位依次排列，为使 <explode_bomb > 函数不被调用，那么内存中存储的任意一个输入数都应满足其大小为相邻低位地址输入数的 2 倍，为相邻高位地址输入数的 $\frac{1}{2}$，即输入的数字构成一个公比为 2 的等比序列, 此为 Phase4 输入满足的第二个条件

2 Phase 5

2.1 条件 1：数量限制

Phase5 函数中共有两个可能导致爆炸的跳转语句，针对其中第一处跳转语句，选取以该语句结束的部分汇编代码进行分析。

选取汇编代码如下：

```
104a8:  2bf000ef            jal ra,10f66 <sscanf>
104ac:  87aa                mv  a5,a0
104ae:  fef42623            sw  a5,-20(s0)
104b2:  fec42783            lw  a5,-20(s0)
104b6:  0007871b            sext.w  a4,a5
104ba:  4789                li  a5,2
104bc:  00f70463            beq a4,a5,104c4 <phase_5+0x40>
104c0:  d65ff0ef            jal ra,10224 <explode_bomb>
```

根据前文可知，执行 <sscanf > 函数后 a0 寄存器中会保存 NumOfNums 值。继续执行代码，在执行判断语句前，已知 a4=NumOfNum 且 a5=2。要使炸弹不爆炸，应有 a4=a5 即 NumOfNums=2 成立。

因此可确定 Phase5 输入满足的第一个条件：输入数字的个数为 2.

2.2 条件 2：关系限制

确定第一个条件后，我们选取两个可能跳转 <explode_bomb > 语句之间的汇编代码进行分析。

具体选取代码如下：

```
104c4:  fe842783            lw  a5,-24(s0)
104c8:  853e                mv  a0,a5
104ca:  0a4000ef            jal ra,1056e <func4>
104ce:  87aa                mv  a5,a0
104d0:  fef42623            sw  a5,-20(s0)
104d4:  fe442703            lw  a4,-28(s0)
104d8:  fec42783            lw  a5,-20(s0)
104dc:  2781                sext.w  a5,a5
104de:  00e78463            beq a5,a4,104e6 <phase_5+0x62>
104e2:  d43ff0ef            jal ra,10224 <explode_bomb>
```

简单转换为伪代码后得到

```
a5=Memory[s0-24]
a0=a5
func4()
a5=a0
Memory[so-20]=a5
a4=Memory[s0-28]
a5=Memory[s0-20]
if(a5!=a4){
    explode_bomb()
}
end
```

要使 <explode_bomb > 不被调用，在进行 if 判断时应有 a4=a5 成立。但因为函数 func4 的存在，寄存器 a0，a4，a5 中保存的值都可能在 func4 函数执行过程中发生改变，所以我们要对 func4 函数进行分析，并以此确定进行 if 判断时寄存器 a4，a5 内的值。

选取 func4 函数汇编代码关键部分：

```
    10576:  87aa                mv  a5,a0
    10578:  fef42623            sw  a5,-20(s0)
    1057c:  fec42783            lw  a5,-20(s0)
    10580:  2781                sext.w  a5,a5
    10582:  e399                bnez    a5,10588 <func4+0x1a>

    10584:  4785                li  a5,1
    10586:  a839                j   105a4 <func4+0x36>


    10588:  fec42783            lw  a5,-20(s0)
    1058c:  37fd                addiw   a5,a5,-1
    1058e:  2781                sext.w  a5,a5
    10590:  853e                mv  a0,a5
    10592:  fddff0ef            jal ra,1056e <func4>

    10596:  87aa                mv  a5,a0
    10598:  873e                mv  a4,a5
    1059a:  fec42783            lw  a5,-20(s0)
    1059e:  02e787bb            mulw    a5,a5,a4
    105a2:  2781                sext.w  a5,a5
    105a4:  853e                mv  a0,a5
```

明显看出 func4 函数以寄存器 a0 中的值作为传入参数，并把计算结果作为返回值保存在 a0 中。将汇编转化为伪代码：

```
a5=a0
Memory[s0-20]=a5
if(a5==0)
{
    a5=1
    a0=a5
    end
}
else if(a5!=0)
{
    a5=Memory[s0-20]
    a5=a5-1
    a0=a5
    func4(a0)
    a5=a0
    a4=a5
    a5=Memory[s0-20]
    a5=a5*a4
    a0=a5
}
end
```

设调用 <func4 > 函数时寄存器 a0 中保存的数值为 n, 结束调用时保存在 a0 中的返回值为 func4(n), 则根据伪代码可以得到如下数学推导式，

$$func4(n) = \begin{cases} n \times func4(n-1) & ,n \geq 2 \\ 1 & ,n = 1 \end{cases}$$

根据数学知识可知，func4(n)=n!。

因此，函数 <func4 > 的作用是以 a0 寄存器中值作传入参数，计算该值的阶乘作为返回值保存到寄存器 a0 中。

已知 func4 函数的输入及输出，将上文的 Phase5 伪代码进行简化得到：

```
1       a5=Memory[s0-24]
2       a5=func4(a0)
3       a4=Memory[s0-28]
4       if(a5!=a4)
5       {
6           explode_bomb();
7       }
8       end
```

设 Phase5 输入的两个数字按顺序分别为 x，y，则根据程序代码 x，y 应满足 $y = x$ 方符合题目要求。

因此可确定 Phase5 输入满足的第二个条件：输入的第二个数等于输入第一个数的阶乘。