一、 已知某个采用 Tomasulo 算法实现乱序执行的处理器信息如下：

- 该处理器的前端分为 fetch 和 decode 两个阶段，fetch 和 decode 操作都只需要 1 个时钟周期
- 该处理器只能执行寄存器类型的指令，例如，OP R_dest <- R_src1, R_src2
- 该处理器每个周期按照指令顺序发送 1 条指令到保留站，指令发送在 decode 阶段完成
- 一条指令被发送到某功能单元的保留站时，按照从上到下的顺序分配保留站空位（如果有空闲的话）
- 当保留站中的指令完成时，将该指令从保留站删除
- 加法和乘法都没有实现 pipeline，分别需要 2 个周期和 3 个周期完成
- 加法和乘法的结果将在 writeback 阶段在总线上广播，需要这些结果的指令最早可以在 writeback 之后的下个时钟周期开始执行（如果所有的操作数都准备好了）
- 当某个功能单元有多个指令都满足可以执行的条件时，按照指令发送的顺序选择最早的指令开始执行

初始时，该处理器上没有任何指令。已知有 4 条指令按照顺序分别发送到了相应的保留站，当第 4 条指令发送完后寄存器和保留站的状态如下图(tag 表示要提供该数据的保留站 ID，0 表示数据已经 ready)：

**寄存器表**

| Reg | Tag | Value |
|---|---|---|
| R0 | - | - |
| R1 | A | 5 |
| R2 | 0 | 8 |
| R3 | E | - |
| R4 | B | - |
| R5 | - | - |

**加法保留站**

| ID | Tag | Value | Tag | Value |
|---|---|---|---|---|
| A | D | - | 0 | 8 |
| B | A | - | A | - |
| C | - | - | - | - |

**乘法保留站**

| ID | Tag | Value | Tag | Value |
|---|---|---|---|---|
| D | 0 | 5 | 0 | 5 |
| E | A | - | B | - |
| F | - | - | - | - |

1. 按照顺序给出发射的 4 条指令分别是什么（格式：OP R_dest <- R_src1,

R_src2）？（7 分）
答案：
MUL R1 <- R1, R1
ADD R1<- R1, R2
ADD R4 <- R1, R1
MUL R3 <- R1, R4

第一条 MUL 的目标寄存器也可以是 R3 和 R4，后面的 ADD 指令也相应修改一下即可。

2. 画出 12 个周期结束后寄存器和保留站的状态。（8 分）

答案：

**寄存器表**

| Reg | Tag | Value |
|---|---|---|
| R0 | - | - |
| R1 | 0 | 33 |
| R2 | 0 | 8 |
| R3 | E | - |
| R4 | 0 | 66 |
| R5 | - | - |

**加法保留站**

| ID | Tag | Value | Tag | Value |
|---|---|---|---|---|
| A | - | - | - | - |
| B | - | - | - | - |
| C | - | - | - | - |

**乘法保留站**

| ID | Tag | Value | Tag | Value |
|---|---|---|---|---|
| D | - | - | - | - |
| E | 0 | 33 | 0 | 66 |
| F | - | - | - | - |