P3 思考题

1.若 PC（程序计数器）位数为 30 位，试分析其与 32 位 PC 的优劣。

首先看一下 PC 的原理。程序计数器需要指示指令地址，指令地址如果按照 4GB 算的话，就是 2^32 字节，需要 32 位 PC；但是每条指令都是 32 位，4 字节，按字对齐，因此最多存放 2^30 条不同指令。所以 PC 也可以设置为 30 位，但这样做需要把地址左移两位才能指向正确的地址。

综上，32 位 PC 可以直接指出地址位置，但存储指令少，而 30 位可以保存更多条指令，但需要移位器特别处理，各有利弊。

2.现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 使用寄存器，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出。

合理。IM 只用读取指令，并不需要写入数据，所以要 ROM。

DM 又要读入数据，又要写入数据，所以要用 RAM。

GRF 使用 32 个寄存器方便对每个寄存器进行写入和读出操作。

3.结合上文给出的样例真值表，给出 RegDst， ALUSrc， MemtoReg，RegWrite, Branch, ExtOp 与 op 和 func 有关的布尔表达式（表达式中只能使用“与、或、非”3 种基本逻辑运算。）

```
 EXTOp[0]                                                =
op[0]&op[1]&!op[2]&!op[3]&!op[4]&op[5]+
op[0]&op[1]&!op[2]&
op[3]&!op[4]&op[5]+  !op[0]&!op[1]&op[2]&!op[3]&!o
p[4]&!op[5]
 EXTOp[1] = op[0]&op[1]&op[2]&op[3]&!op[4]&!op[5]
 MemtoReg                                                =
op[0]&op[1]&!op[2]&!op[3]&!op[4]&!op[5]
 MemWrite = op[0]&op[1]&!op[2]&op[3]&!op[4]&op[5]
Branch = !op[0]&!op[1]&op[2]&!op[3]&!op[4]&!op[5]
```

```
 ALUOp[0]
=  !op[0]&!op[1]&!op[2]&!op[3]&!op[4]&!op[5]&fun[0
]&fun[1]&!fun[2]&!fun[3]&!fun[4]&fun[5]+  !op[0]&!
op[1]&op[2]&!op[3]&!op[4]&!op[5]
 ALUOp[1]
=        !op[0]&!op[1]&!op[2]&!op[3]&!op[4]&!op[5]
+op[0]&op[1]&!op[2]&!op[3]&!op[4]&!op[5]+
op[0]&op[1]&!op[2]&op[3]&!op[4]&op[5]
 ALUOp[2] = op[0]&op[1]&op[2]&op[3]&!op[4]&!op[5]
 ALUSrc = op[0]&!op[1]&op[2]&op[3]&!op[4]&!op[5]+
op[0]&op[1]&!op[2]&!op[3]&!op[4]&op[5]+
op[0]&op[1]&!op[2]&op[3]&!op[4]&op[5]+
op[0]&op[1]&op[2]&op[3]&!op[4]&!op[5]
 RegDst
= !op[0]&!op[1]&!op[2]&!op[3]&!op[4]&!op[5]
 RegWrite
=       !op[0]&!op[1]&!op[2]&!op[3]&!op[4]&!op[5]+
op[0]&!op[1]&op[2]&op[3]&!op[4]&!op[5]+
op[0]&op[1]&!op[2]&!op[3]&!op[4]&!op[5]+
op[0]&op[1]&op[2]&op[3]&!op[4]&!op[5]
```

4.充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式， 请给出化简后的形式。

同上题

5.事实上，实现 nop 空指令，我们并不需要将它加入控制信号真值表，为什么？请给出你的理由。

当 32 位指令都为 0 时，我们执行写入寄存器操作，写入 0 号寄存器，并不会造成影响。

6.前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM 片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移。

不需要修改，因为我们取的是地址的后 10 位（实际上是 5 位），无论起始地址设置为 0x0000 还是 0x3000 均被截断高位，起始地址自然会成为 0。对于偏移量不能跨地址的情况，可以把所有的 address 全部右移两位。这样，无需任何操作便可以正常运行。

7.除了编写程序进行测试外，还有一种验证 CPU 设计正确性的办法——形式验证。 形式验证的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证（Formal Verification)"了解相关内容后，简要阐述相比与测试，形式验证的优劣。

形式验证是一个系统性的过程，将使用数学推理来验证设计意图（指标）在实现(RTL)中是否得以贯彻。形式验证可以克服所有 3 种仿真挑战， 由于形式验证能够从算法上穷尽检查所有随时间可能变化的输进值。

对组合逻辑来说，不存在状态寄存器，其输出值 Z[t]不依赖于前面的输入值 X[t-i](1≤i≤t)。这时只要对每个输入向量证明其输出向量相同。

对一个时序电路而言，可以把它看成一个有限状态机(FSM，finite-state machine)。电路功能的等价可以用有限状态机的等价来判断。假定有两个状态机 A 和 B，要对它们进行比较。直观的说，当 A 和 B 有相同的接口，而且从相同的初始状态出发，两者对有效输入值序列产生相同的输出值序列，则可以说 A 和 B 等价。

形式验证的优点如下：

(1)形式验证是对指定描述的所有可能的情况进行验证，覆盖率达到了 100%。

(2)形式验证技术是借用数学上的方法将待验证电路和功能描述或参考设计直接进行比较，不需要开发测试激励。

(3)形式验证的验证时间短，可以很快发现和改正电路设计中的错误，可

以缩短设计周期。

缺点是验证方法复杂抽象，难以准确把握。而且形式验证是数学逻辑分析，而不是电路分析，不能有效的验证电路的性能，如电路的时延和功耗等。