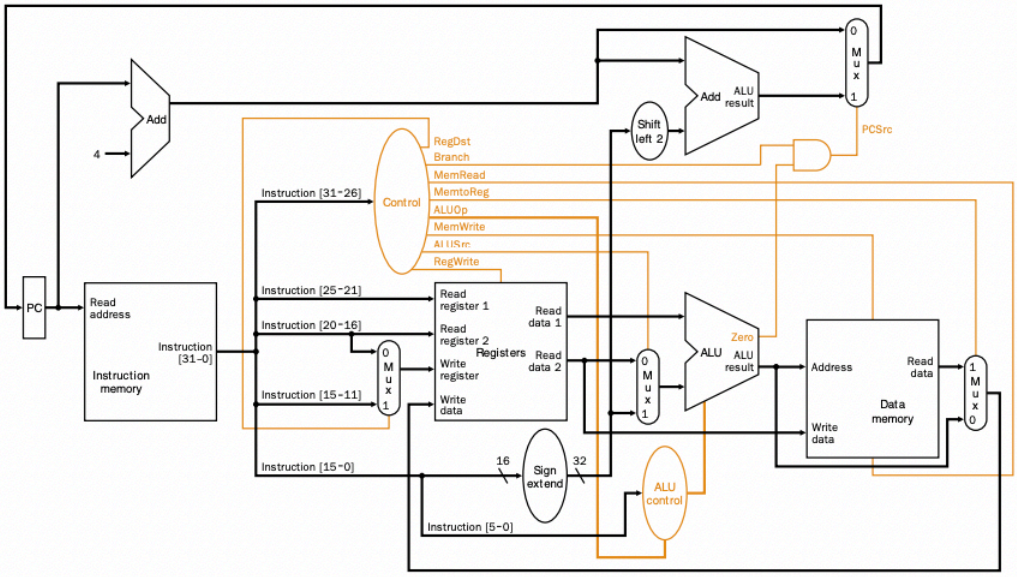
《计算机体系结构课程》作业 2

作业 2（1）

Modify the single-cycle processor datapath to include a version of the `lw` instruction, called `lw2`, that adds two registers to obtain the effective address. The datapath that you will modify is provided below. Your job is to implement the necessary data and control signals to support the new `lw2` instruction, which we define to have the following semantics:

`lw2`: Rd ← Memory[Rs + Rt]
PC ← PC + 4

Add to the datapath any necessary data and control signals (if necessary) to implement the `lw2` instruction. Draw and label all components and wires very clearly (give control signals meaningful names; if selecting a subset of bits from many, specify exactly which bits are selected; and so on).



ALU opcode	Operation
00	Add
01	Subtract
10	Controlled by `funct`
11	Not used

作业 2（2）

一个多周期处理器 P1 执行读内存（Load）指令需要 10 个周期，写内存（Store）指令需要 8 个周期，算术指令需要 4 个周期，分支指令需要 4 个周期。考虑一个应用程序 A，其中 20%的指令是 Load 指令，20%的指令是 Store 指令，50%的指令是算术指令，10%的指令是分支指令。

（1）在处理器 P1 上执行 A 程序，CPI（cycles per instruction）是多少？请写明计算过程。

（2）新的处理器设计 P2 将 P1 的时钟频率加倍，但是 Load、Store、算术和分支指令的延迟分别增加了 2、2、2 和 1 个周期。P1 和 P2 使用相同的编译器，程序 A 在 P1 和 P2 上编译后生成相同数量的指令。请计算在 P2 处理器上执行应用程序 A 时，CPI 是多少？请写明计算过程。

（3）程序 A 在哪个处理器上运行更快？快多少倍？请写明计算过程。

（4）假设在处理器 P1 的芯片中有一些额外的空间，可以容纳额外的硬件。备选方案一是在 P1 中加入一个更快的分支执行单元，该分支执行单元能将分支指令的延迟缩短 4 倍；备选方案二是在 P1 中加入一个更快的存储设备，该存储设备能将内存相关操作的延迟缩短 2 倍。请结合阿姆达尔定律（Amdahl's Law）说明应该选择哪种备选方案，并写明计算过程。

作业 2（3）

假设我们测试两个处理器 A 和 B 在基准程序上的性能。我们发现每个处理器的情况如下：

- 处理器 A 的 CPI 为 2，每秒执行 40 亿条指令。
- 处理器 B 的 CPI 为 1，每秒执行 80 亿条指令。

哪个处理器在这个程序上的性能更高？请回答并说明理由。

注：CPI 代表每条指令的周期数。

作业 2（4）

假设有输入为A_0、A_1、A_2、A_3，输出为X和Y的逻辑函数，请根据如下布尔方程，填写真值表

$$X = (A_3 + \overline{A_2} + \overline{A_1} + \overline{A_0}) \cdot (\overline{A_3} + \overline{A_2} + \overline{A_1} + A_0) \cdot (\overline{A_3} + \overline{A_2} + \overline{A_1} + \overline{A_0})$$

$$Y = \overline{A_3} A_2 \overline{A_1} A_0 + A_3 \overline{A_2} A_1 \overline{A_0}$$

Inputs				Outputs	
A_3	A_2	A_1	A_0	X	Y
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		