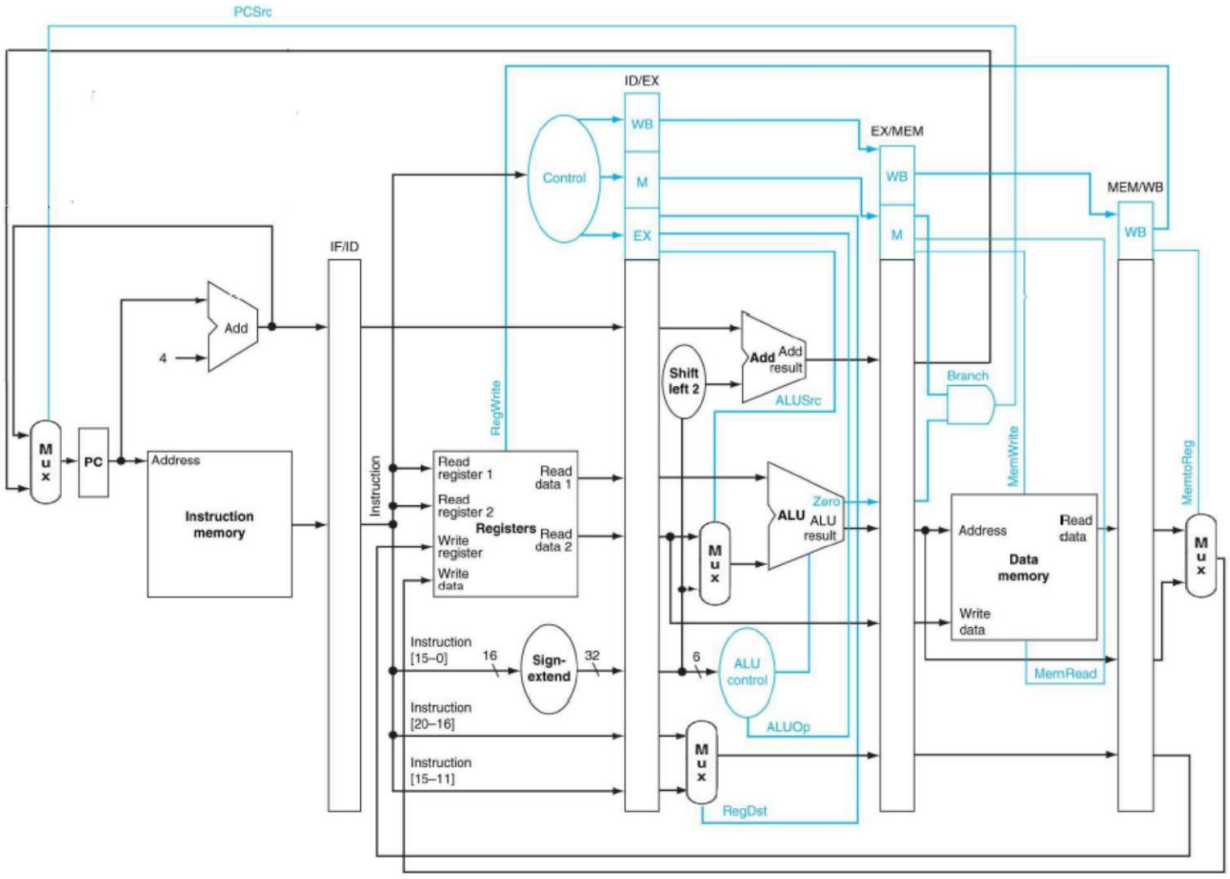
Computer Organization

Architecture diagram:



Detailed description of the implementation:

1. **Pipe_CPU_1.v**
   這次為了加入 Pipeline 的架構，我們依照上次實作的 Simple_cpu，在中間加上了 Pipeline Reg，將每個 Pipeline Register 接上每個 module 的 input 與 output，讓 Pipeline Register 能夠把每個 Stage 的 Control Signal 傳入下個 Stage。根據每個 Signal 使用時機的不同，接入不同的 Register: WB、M、EX。

2. **Reg_File.v**
   為了處理同一個瞬間讀取並寫入相同的 Register，將 Reg_File 的 Clock 改為 negedge，讓讀取的時間點延後半個 Clock，才可以得到正確的值。

Finished part:

Basic part, without hazard detection and forwarding.

Problems encountered and solutions:

由於需要建的線實在是很多，需要仔細看每一條線對應到的 Pipeline Reg 以及是 input 還是 output，否則如果接錯的話，可能需要花很多時間 Debug，而且在 Pipeline 的結構下，每個 Stage 都會相差一個 Clock，想要 Debug 更是困難。

Summary or Lesson learnt (if any):

我學到在波形圖中，有如茫茫大海的 0 和 1 之間，找出我想要的 0 和 1，這 過程有如重度近視的人在地板上找一根針，像個迷途羔羊常常來回比對迷失了方向，我想這就是這作業迷人的地方吧。這次的作業我覺得最困難的地方在於由於加上了 pipeline，有些 control signal 會延遲幾個 clock，導致在 Debug 的時候無法直接在同一個 clock 判斷是否正確執行，增加了 debug 的難度。雖然很費工夫，但也更了解整個 Pipeline CPU 的架構，算是有所成長。