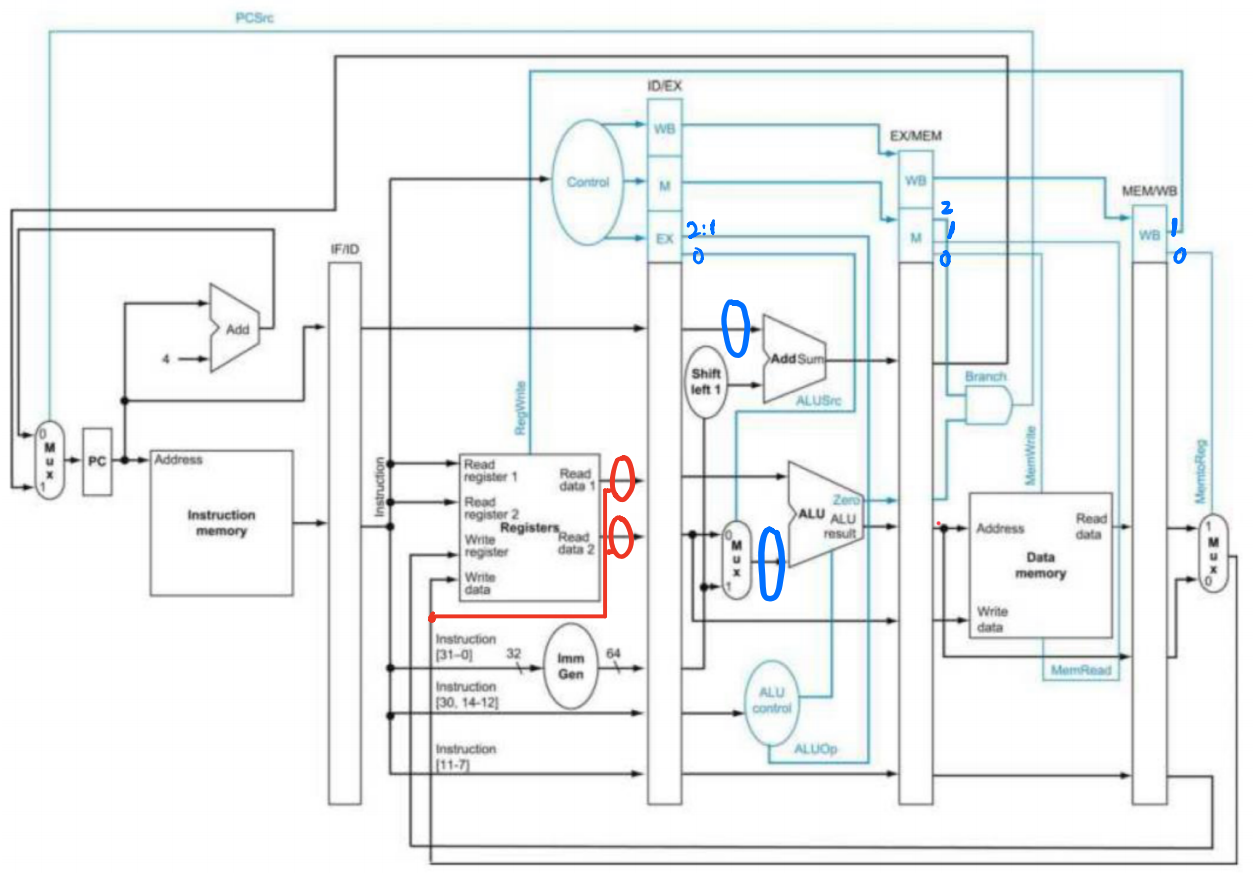
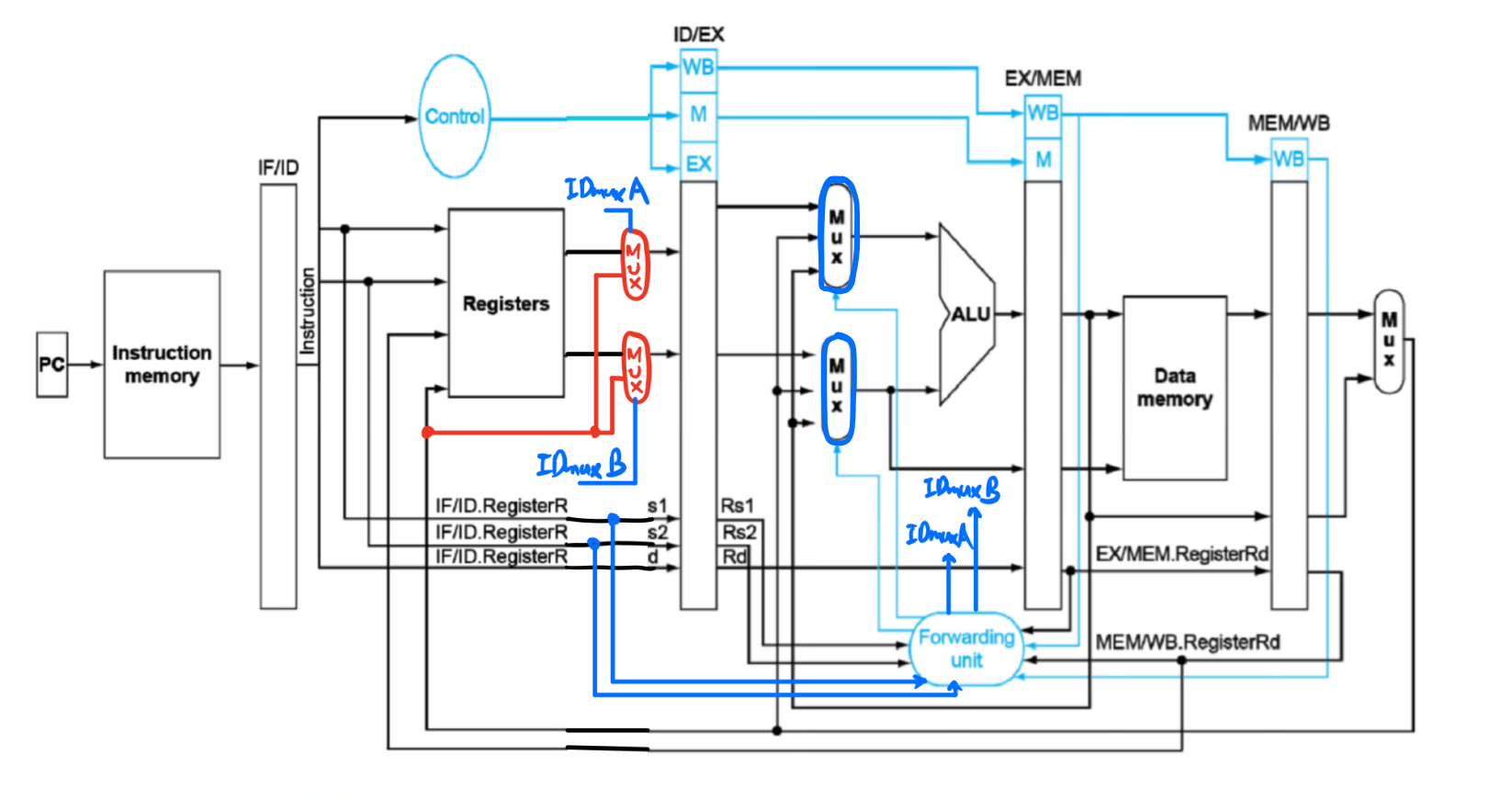
**Computer Organization**

**Architecture diagram:**

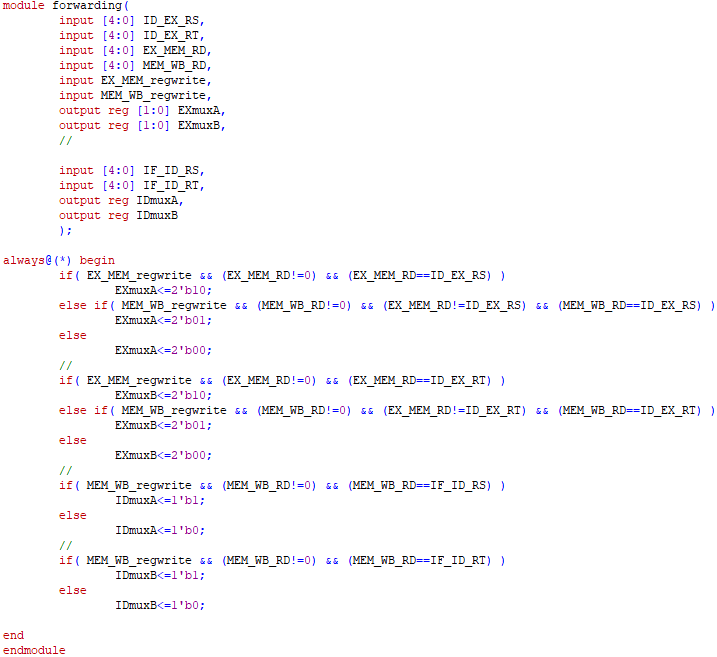
****

**forwarding部分:**

****

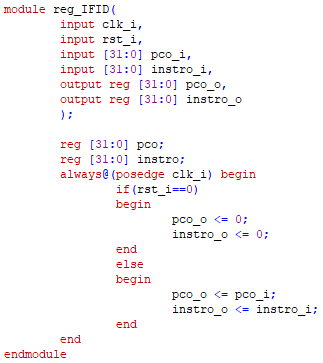
**Detailed description of the implementation:**

**forwarding:**

****

**forwarding按課本所述接，再多偵測MEM/WB rd 跟 IF/ID rt,rs是否相同。因為寫入Register需要一段時間，若寫入同時即需讀取會來不及，所以多用兩個MUX(架構圖中紅色部分)提前取得寫入的值。**

**分隔pipeline的register:**

****

**四個register的架構都大致相同。**

**Control(Decoder):**

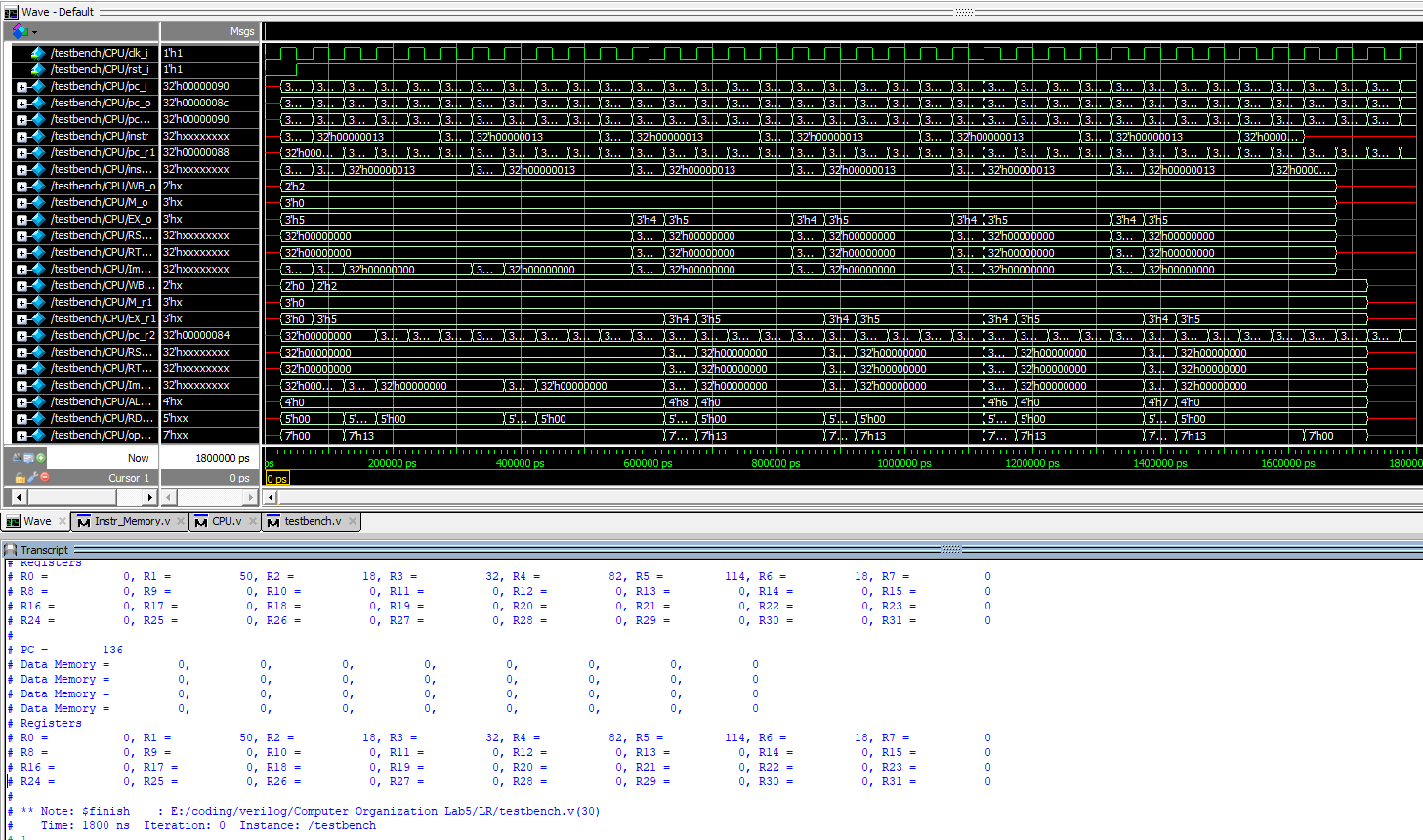


**我們將decoder(現control)的輸出合併成ID,EX,MEM跟WB，**

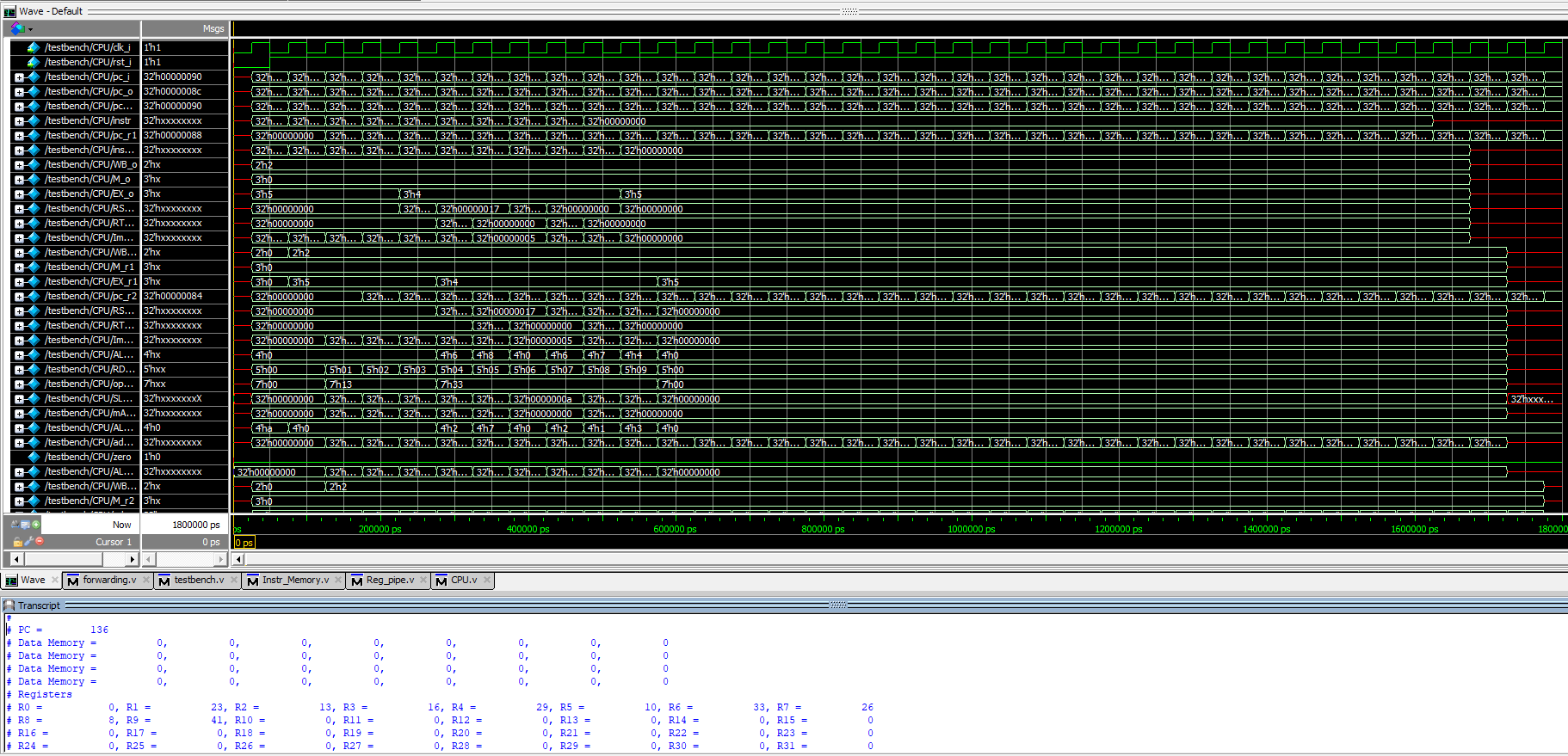
**其餘模組都照舊。**

**Implementation results:**

**test\_data1:**



**test\_data2:**

****

**Problems encountered and solutions:**

**因為這次要加入pipeline，單是所需要接的線就多了將近一倍，預先將架構圖畫出就顯得相當重要。**

**而且此架構與single cycle的CPU有顯著的不同，如何將pipeline切開的各部分充分利用就產生了許多技術上的困難，如register寫入和讀取的間隔。並非單靠課本上所教的forwarding就能解決。**

**在寫CPU架構的時候，為求整齊而將模組按pipeline分開也讓後來在debug的時候能夠快速地找到問題，省去了許多的麻煩。**

**CPU只會越來越複雜，在動手打code之前，有計畫性地事先規劃便是我們這次學到最重要的課題。**