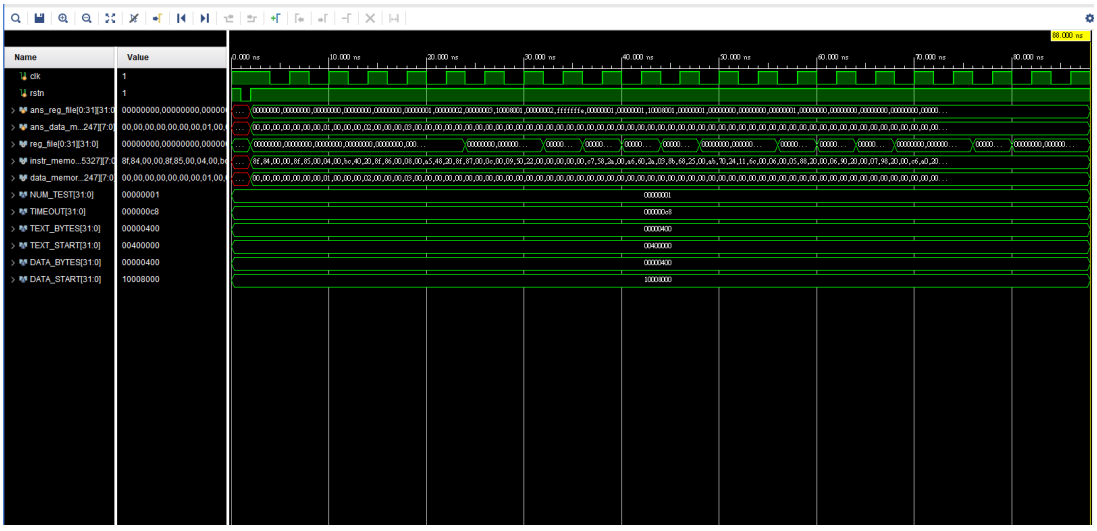
Lab 4: Pipelined Processor - Part 2

學號: 111550129　姓名: 林彥亨

1. (1%) Experimental Result
   a. (1%) Show the waveform screen shot of the test we provided.



   b. (1%) What other cases you've tested? Why you choose them?

2. (8%) Answer the following Questions
   a. (2%) List out the equation to detect EX & MEM hazard in forwarding unit. Which part of the equation in textbook p.369 is wrong?

   EX/MEM Hazard=
   (*ID/EX.RegRs=EX/MEM.RegRd*)∨(*ID/EX.RegRt=EX/MEM.RegRd*)

   2. *MEM hazard:*

```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠  0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠  0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
```

b. (2%) In forwarding for beq , is forwarding from MEM/WB to ID needed? Why?

   Forwarding from MEM/WB to ID for `beq` is typically not needed if the register file update happens before the decode stage of `beq`. However, specific pipeline designs or timing issues (like late write-backs in the cycle) may necessitate this type of forwarding to ensure correct and timely availability of data for branch decision-making. This scenario becomes critical in very tightly timed or deeply pipelined processors.

c. (2%) Briefly explain how you insert 2 stalls when beq reads registers right after lw writes it.

   By adding these two stalls, the pipeline ensures that `beq` accesses the correct, updated values from the register, resolving the data hazard and maintaining correct program flow. These stalls are critical for avoiding incorrect branch decisions based on outdated or wrong register data.

d. (2%) sw right after lw is quite common since copy and paste a data from one address to another is used frequently. In textbook, a stall is followed by the lw in this case. Is it possible to remove this stall? How?

   To remove the stall that typically occurs between a `lw` (load word) instruction and a subsequent `sw` (store word) instruction, you can use forwarding. This technique involves forwarding the data from the MEM stage of the `lw` instruction directly to the ID stage of the `sw` instruction. By implementing this data bypass, the `sw` instruction can access the needed data immediately after it's fetched by `lw`, without waiting for the WB stage, thereby eliminating the stall. This assumes the hardware supports such forwarding capabilities.