Report for Lab5(1 day late token)

In this lab, we need to implement forwarding and hazard detection. The compilation order:

1. IMEM outputs the instructions corresponding to the address according to the address sent by the PC. Whether to write this instruction depends on whether a hazard is detected. If a hazard occurs, the instruction writing is stopped, which is the first stage.

2. The IFID register transfers instructions to CPUcontrol, Hazard detector, register, and IDEX register. The address of the operation register and destination register is given to the IDEX register for subsequent forward comparison.

3. IDEX transmits the data of the corresponding register to two multiplexers, and then selects the data input to ALU according to the signal given by forwarding. If the address of the register is the same as ALUresult, the result will be returned as the input of ALU.

4. Then store the data in the corresponding memory address, and transfer the data to be written back to the MEMWB register, and then write it back to the corresponding register.

I want to illustrate the correctness of the experiment through the following table.

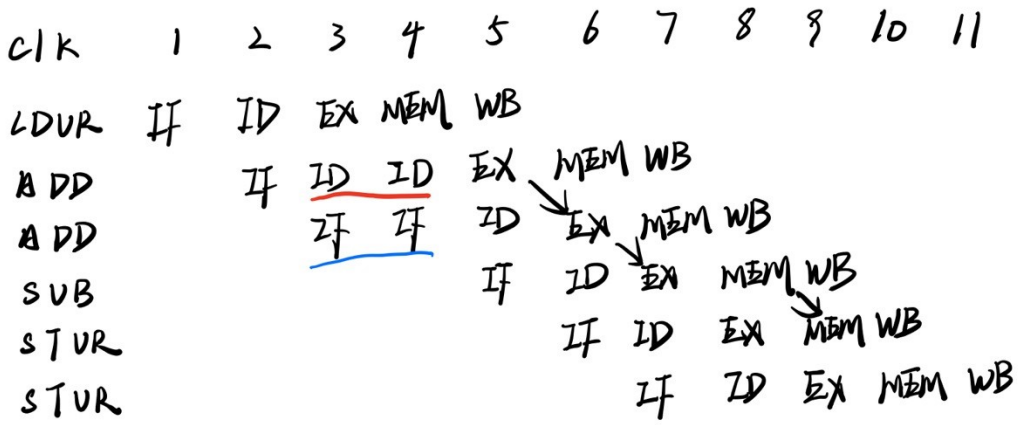

Figure 1.1 Stage Form

According to the first instruction, we need to give the data in the corresponding memory address to X9 in the WB stage, but in the second instruction, we need to execute the addition operation on X9, so a hazard is generated, which is marked by the red underline in the above table between clock 3-4. At the same clock period, the third instruction will stall, which is marked with blue underline.

When we execute the STUR instruction, we need to store the data in X11 in the corresponding memory address. Therefore, according to the above table, due to the data in X11 is 2, it will be stored in the corresponding 8 and 16 addresses respectively.

The black arrow indicates the situation of forwarding. After the calculation result of the previous instruction is obtained, the calculated data is directly handed over to the next instruction for calculation without going through two stages of memory and write back.

Then, we can see from Figure 1.2 that in the third clock cycle, write_enable signal is 0, and a hazard is generated, which is marked in the red area of the figure. The blue rectangle is the stall of the third instruction, and the red arrow indicates that the memory data is written to X9 at the fifth and seventh clock cycles.

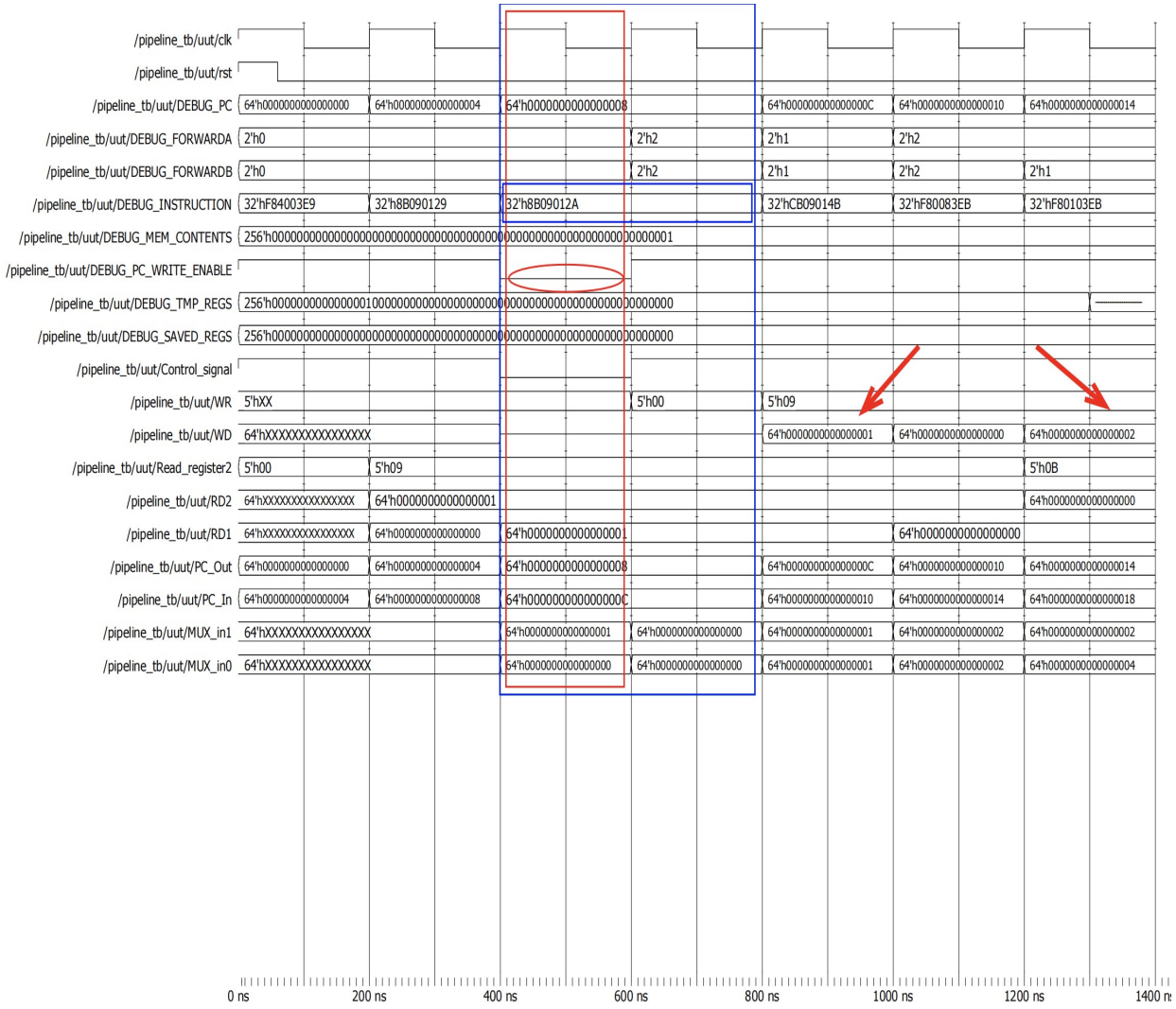

Figure1.2 Hazard and Stall

After executing the subsequent instructions, it can be seen in Figure 1.3 that the data written to the corresponding register is also correct, and the correct values are written in the memory at the corresponding positions of DMEM[8] and DMEM[16].

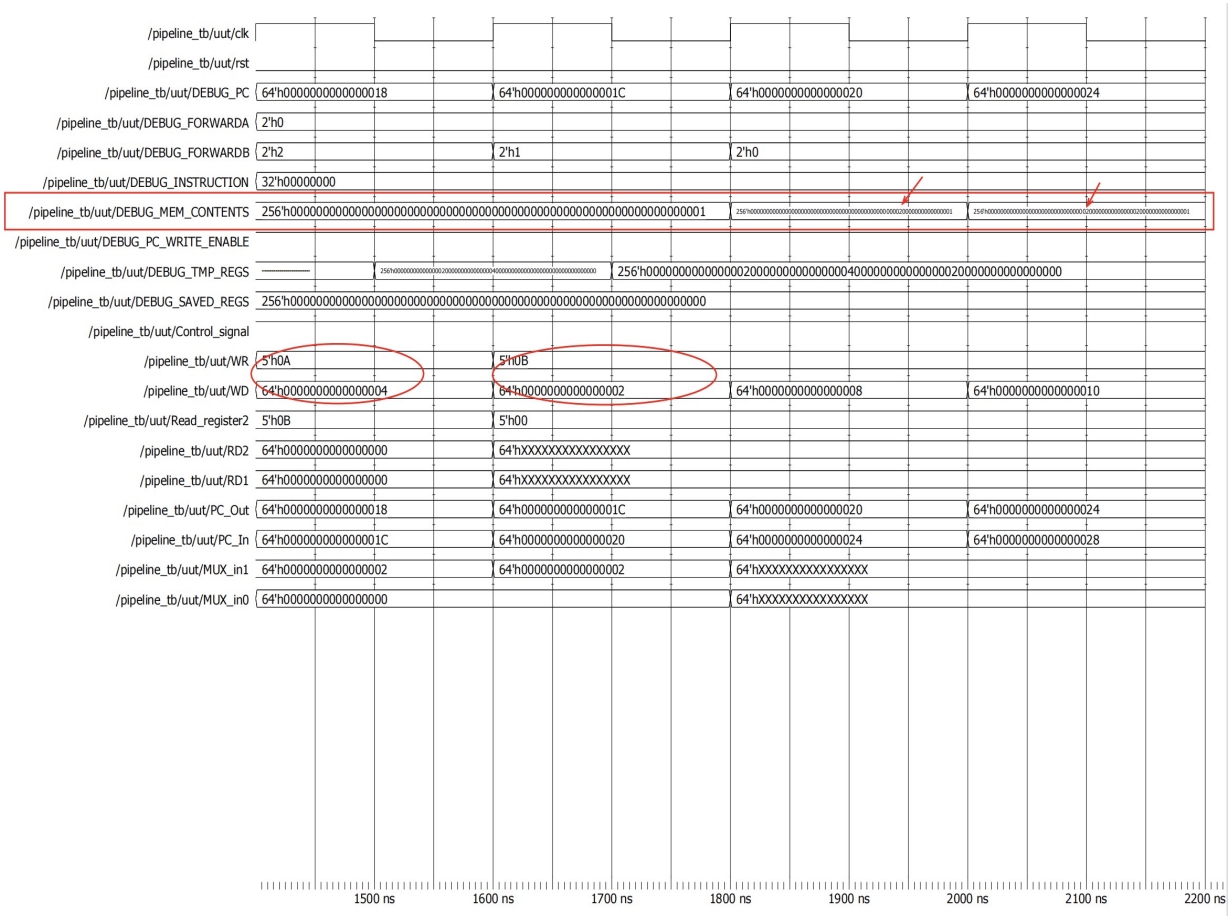

Figure 1.3 MEM_CONTENTS