Report

I want to use two late tokens for lab5

This experiment mainly implements two components, hazard detection and forward unit. These two components allow the processor to run faster, and can be stalled when a hazard is detected, so that subsequent instructions can be run with correct values.

This experiment mainly implements two components, hazard detection and forward unit. These two components allow the processor to run faster, and can be stalled when a hazard is detected, so that subsequent instructions can be run with correct values. .

Hazarddetect accepts the signal and outputs the PCwrite signal to control the output address of the PC section. The IF/ID write signal it outputs controls the output of the IF/ID section. The output selection signal controls the output of CPUcontrol. When the hazard signal is detected, these are 0. The processor will stall.

Forwadunit, accepts the signal, determines the forwardA and forwardB of the output, these two signals will control the input signal of Alu through the selector.

Compilation order:

The PC part outputs the address to the imem part, and the imem part outputs the instruction signal to the IF/ID part. Under normal

circumstances, IF/ID will transmit instructions to registers and Hazarddetect, as well as CPU, ID/EX part. The command signal output from the IF/ID will enter the hazard part, and when the hazard is detected, a control signal will be sent to control the cpucontrol IF/ID and PC part, and the stall will be performed.

In the ID/EX part, the signals output by the register and cpucontrol will enter the ID/EX, and the ID/EX will transmit the signal to enter the two selectors of three options and the Forwarding unit. These two multiplexers are composed of forwardA and forwardB Controlled, transmitted to the ALU component after multiple operations.

Forwarding part, this part controls the output of fowardA and fowardB by receiving the data transmitted by IF/ID, and decides which signal to output to the ALU component by controlling two three-choice selectors.

In the EX/MEM part, the signal output by EX/MEM will be stored in the corresponding memory address. And send the data to be written back to MEMWB, and finally write back to the corresponding register to change the value of the corresponding register (write WD to the address corresponding to WR)

For flowing instructions:

```
LDUR  X9, [XZR, 0]          11111000010000000000001111101001
ADD   X9, X9, X9            10001011000010010000000100101001
ADD   X10, X9, X9           10001011000010010000000100101010
SUB   X11, X10, X9          11001011000010010000000101001011
STUR  X11, [XZR, 8]         11111000000000001000001111101011
STUR  X11, [XZR, 16]        11111000000000010000001111101011
NOP                         filled with zero - see imem
NOP                         filled with zero - see imem
NOP                         filled with zero - see imem
NOP                         filled with zero - see imem
```

Figure 1: the instruction order

The first instruction will read the value of dmem0 into the x9 register, but this should belong to the task of the WB stage. But in the second instruction, X9 needs to be operated directly, because the value of X9 has not been written or changed at this time. Therefore, the processor will detect a datahazard1 corresponding to the selection signal at this time should be 0. The hazard should be detected in the third and fourth cycles of the clock. After the second instruction, X9=2, at the third instruction, x10=4, after the fourth instruction, X11=2, and due to the existence of the forwardunit component, no datahazard will appear. These few instructions and run normally. Then in the fifth and sixth instructions, X11=2 will be stored in the 8th and 16th offset positions respectively, which can be seen by checking the value of dmemcontent after the ninth cycle.

Cycle	1	2	3	4	5	6	7	8	9	10	11
	IF	ID	EX	MEM	WB						
		IF	ID*	ID	EX	MEM	WB				
			IF*	IF	ID	EX	MEM	WB			
					IF	ID	EX	MEM	WB		
						IF	ID	EX	MEM	WB	
							IF	ID	EX	MEM	WB

Hazard is only detected in the third cycle, and the processor will stall in the third instruction.

* denotes the hazard detect and stall

For Extra Credit(I just want to try)

running the test program with a nop inserted after the first instruction

stalling the pipeline: I think there won't be stall in the pipeline because of the nop insertion.

forwarding values:

FowardA: 0-4 cycles=0, 5cycle=1, 6-7cycle=2, 8-end =0

FowardB: 0-4 cycles=0, 5cycle=1, 6cycle=2, 7cycle=1, 8 cycle=2, 9cycle=1. 10cycle-end=0

number of cycles needed to finish the program 11cycles

value of PC when the final sw occurs: 24(stur end)

the state of DMEM/Registers at the end of the program:

X9=2, X10=4, X11=2

DMEM_content= 256h'000000....2.....2....1

Use all 0s for NOPs

* denotes the hazard detect and stall

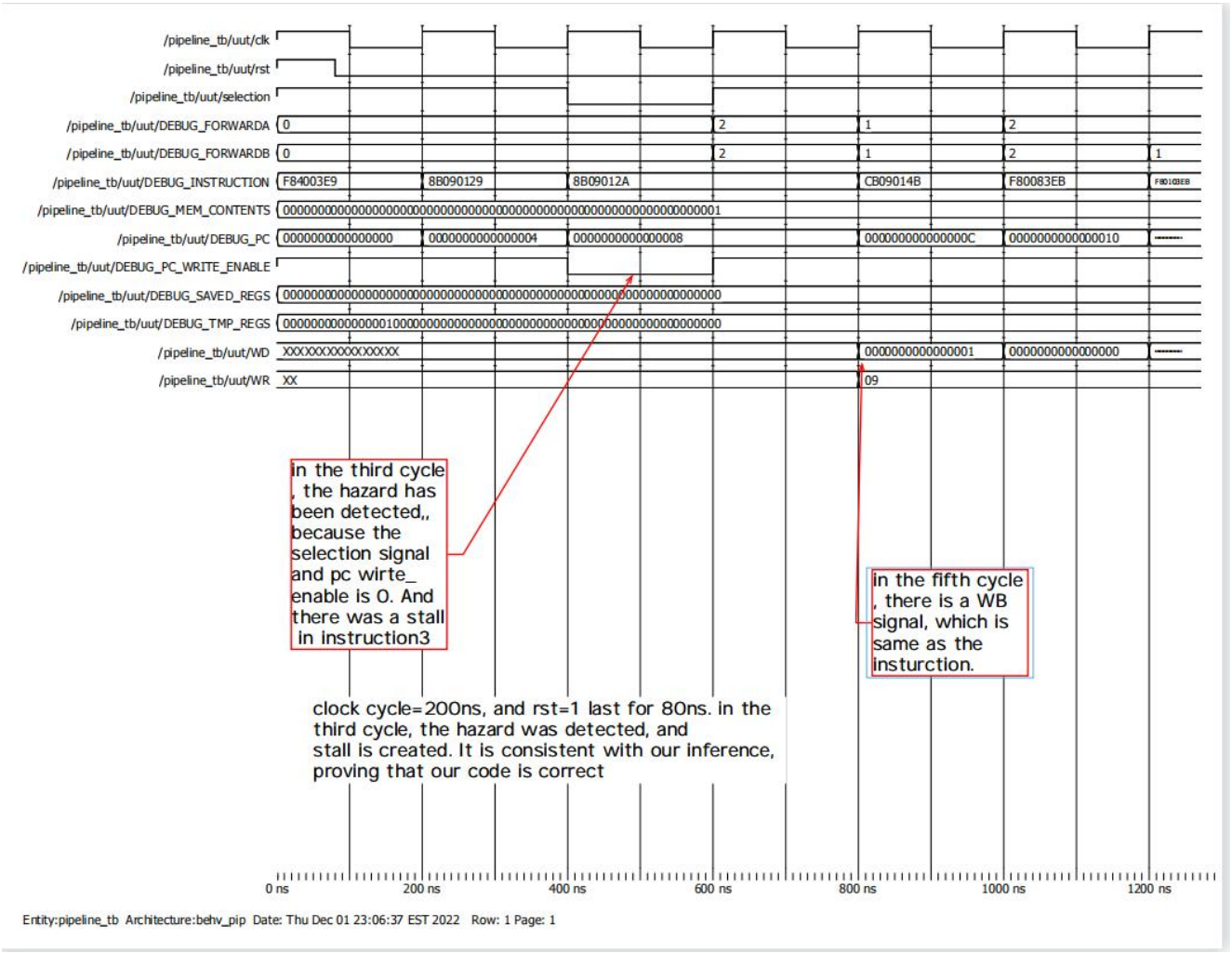

Figure 2: the hazard detection and Stall

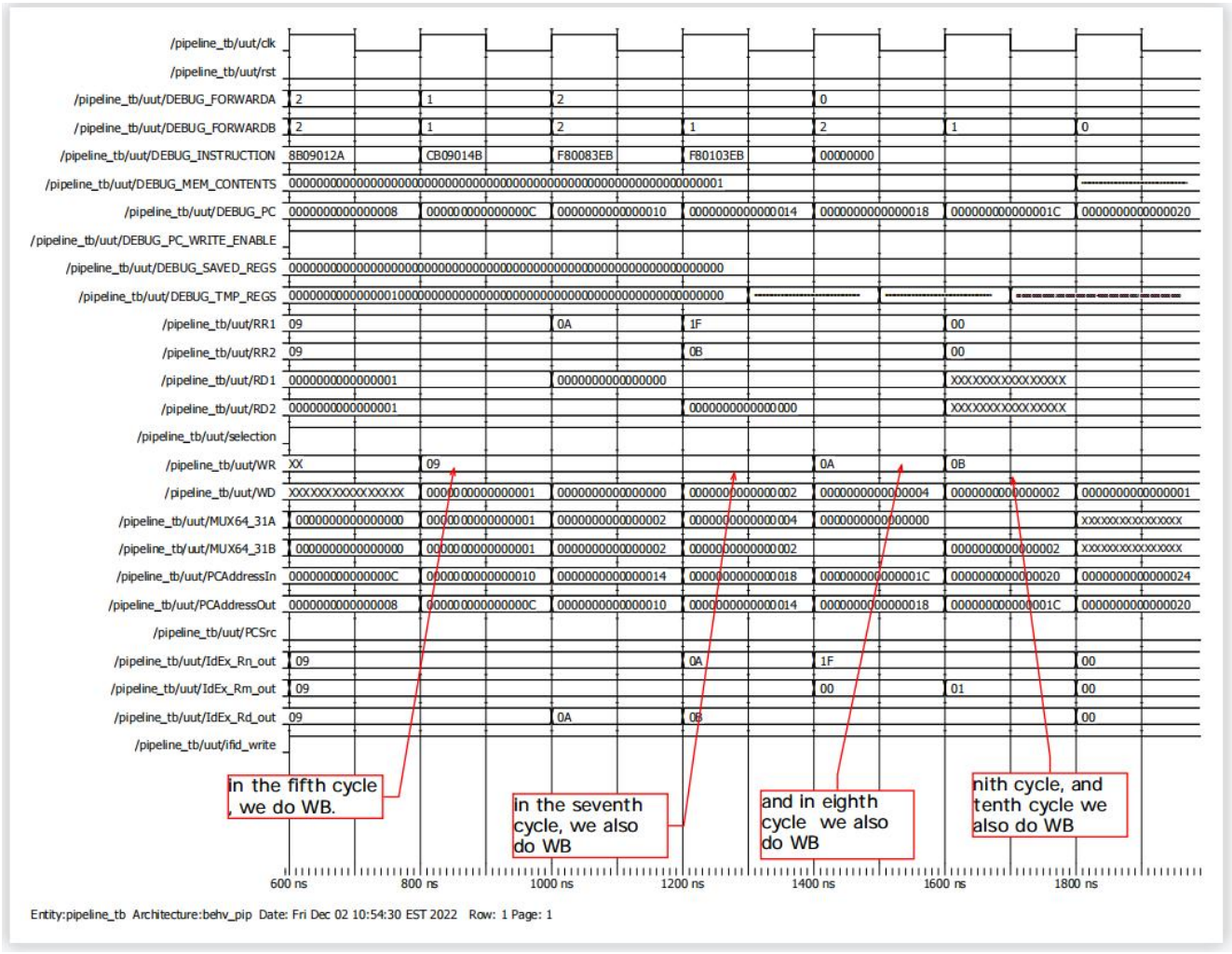

Figure 3: the WB stage shown

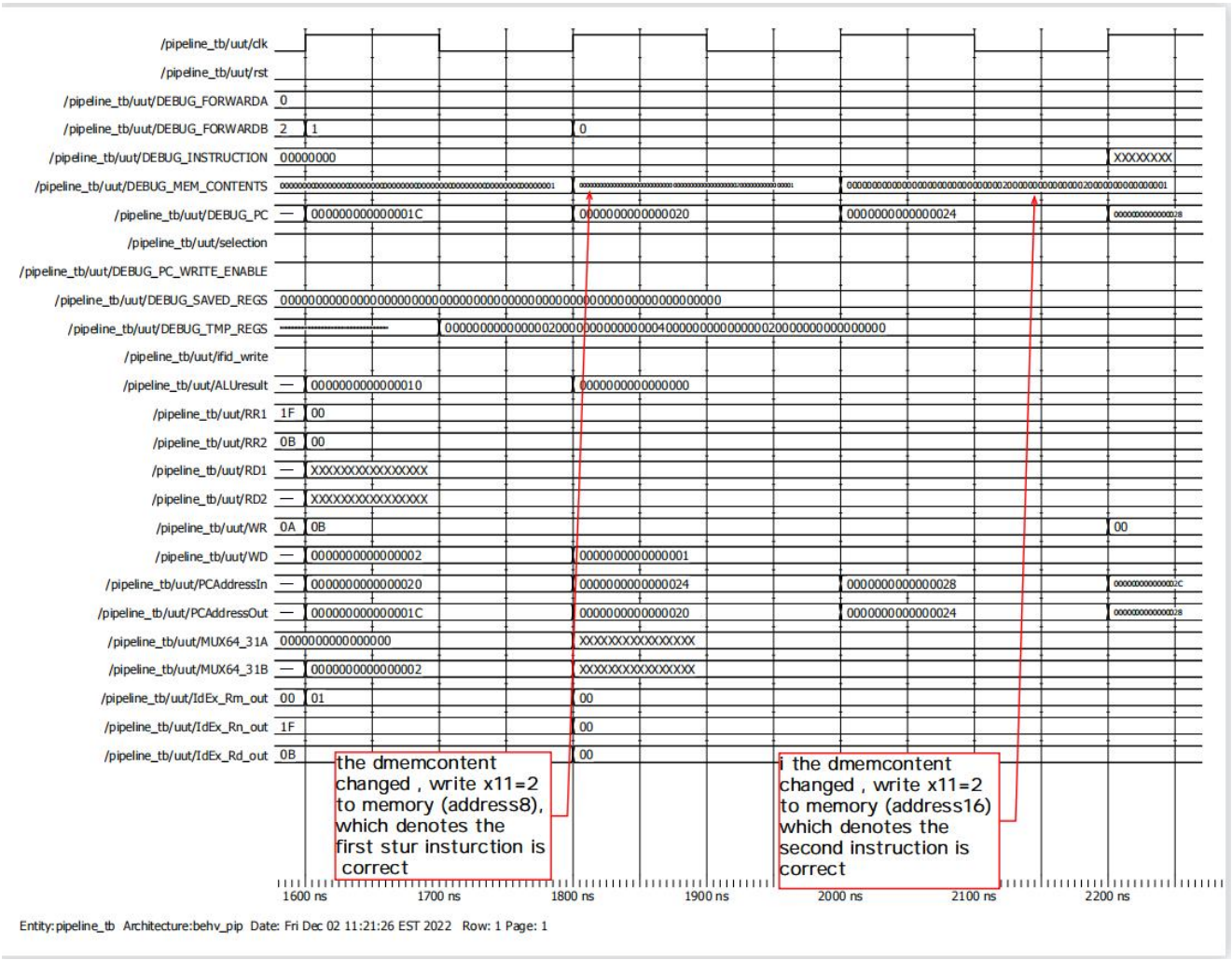

Figure 4: the memory content change