Lab Assignment #5

EE 126: Computer Engineering

Tufts Univerisity, Fall 2019

Name: Ruoxi Ren Student ID: 1327867 E-mail: Ruoxi.Ren@tufts.edu

1 Waveforms of Pipelined CPU without Hazard Detection

The CPU runs a MIPS program as below:

```
lw $t0 0x0($zero)
add $t0 $t0 $t0
add $t1 $t0 $t0
sub $t2 $t1 $t0
sw $t2 0x4($zero)
sw $t2 0x6($zero)
nop // sw -> ID
nop // sw -> EX
nop // sw -> MEM
nop // sw -> WB
```

Before running the program, some registers are set as below:

Registers	Value
$t0	0
$t1	0
$t2	0
$t3	0
$s0	0
$s1	0
$s2	0
$s3	0

And the memory is set as below:

Address	Value
DMEM(0x0)	0x02 02 01 01
DMEM(0x4)	0x00 00 00 00
DMEM(0x8)	0x00 00 00 00
DMEM(0xC)	0x00 00 00 00

The waveform of cycle 1 to cycle 6 is shown in Fig. 1.

In cycle 1, the pipeline is reset. The first instruction is fetched. Since there is a rest signal at the beginning of the cycle, the instruction in ID stage is nop. Besides, all the registers and memory are set to the initial value.

In cycle 2, the second instruction is fetched. The first instruction enters the ID stage, and the control signal is set according to the instruction. It is lw instruction so $RegWrite$ signal is 1. The nop instruction is in the EX stage.

In cycle 3, the third instruction is fetched. The first instruction enters the EX stage while the second instruction is in the ID stage. The second instruction needs the value of $t0 which should be forwarded from MEM stage. However the first instruction is still in the EX stage. So now there is a stall which can be seen from the $DEBUG\_PC\_WRITE\_ENABLE$ signal.

In cycle 4, no instruction is fetched. The first instruction enters the MEM stage while the second instruction is in the ID stage. The third instruction is still in IF stage. There is a nop in EX stage.

In cycle 5, the fourth instruction is fetched. The third instruction enters ID stage while the second instruction enters the EX stage. There is a nop in MEM stage. The second instruction needs the latest value of $t0, so the data from WB is forwarded to the input of ALU, which can be seen from the $DEBUG\_FORWARDA$ and $DEBUG\_FORWARDB$ signal. The first instruction is in WB stage, so in the second half of the cycle, the value of $t0 is changed into 0x02020101.

In cycle 6, the fifth instruction is fetched. The fourth instruction enters ID stage while the third instruction enters the EX stage. There is a nop in WB stage and the second instruction is in MEM stage. However, the third instruction which is in the EX stage needs the latest value of $t0. Thus the value of ALU result from MEM stage is forwarded to the input of ALU, which can be seen from the $DEBUG\_FORWARDA$ and $DEBUG\_FORWARDB$ signal.

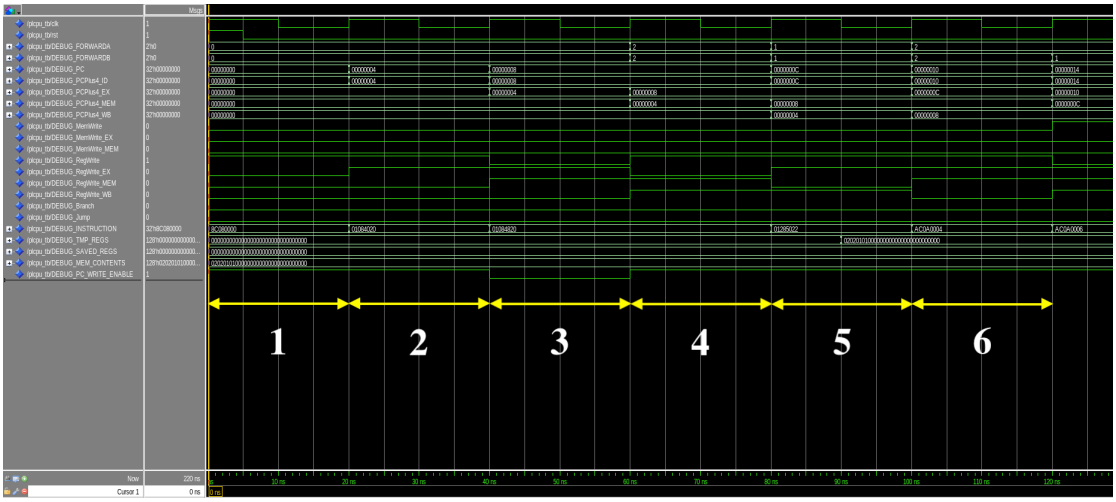

Figure 1: Simulation of pipelined CPU for cycle 1 to 6

The waveform of cycle 6 to cycle 11 is shown in Fig. 2.

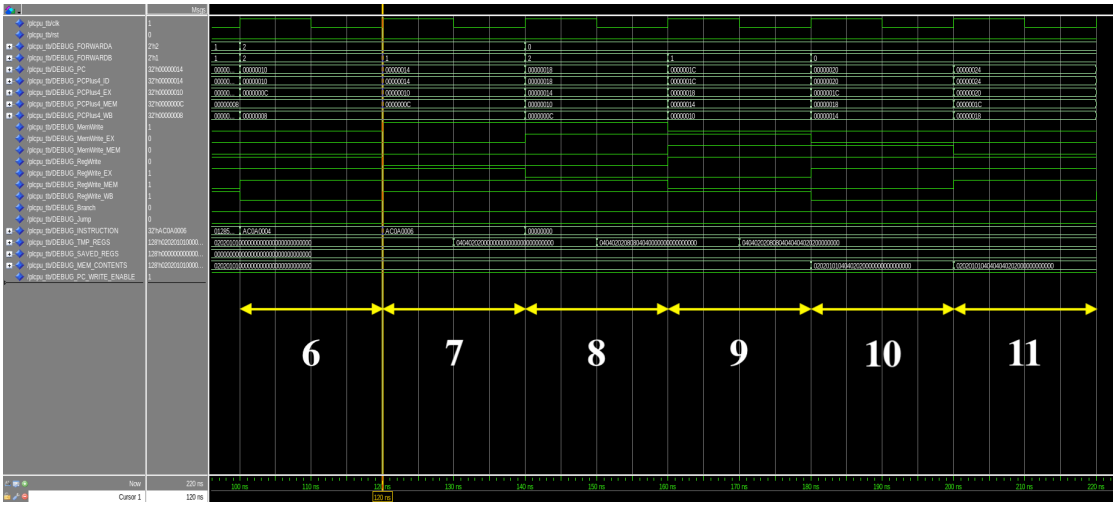

Figure 2: Simulation of pipelined CPU for cycle 6 to 11

In cycle 7, the sixth instruction is fetched. The fifth instruction enters the ID stage while the fourth instruction is in the EX stage. The third instruction is in MEM stage while the second

instruction is in WB stage. However, the forth instruction which is in the EX stage needs the latest value of $t1 and $t0. Thus the value of ALU result from MEM stage and the value of $t0 which has not been written back in WB stage are forwarded to the input of ALU, which can be seen from the $DEBUG\_FORWARDA$ and $DEBUG\_FORWARDB$ signal. Meanwhile, the new value of $t0 is written back, so in the second half of the cycle, the value of $t0 is changed into 0x04040202.

In cycle 8, the nop is fetched. The sixth instruction enters the ID stage while the fifth instruction is in the EX stage. The forth instruction is in MEM stage while the third instruction is in WB stage. However the fifth instruction needs the latest value of $t2 to write to the DMEM, so the value of ALU result from MEM stage is forwarded to the second input of ALU, which can be seen from the $DEBUG\_FORWARDB$ signal, but it is not used by the ALU because the ALU calculates the address of the data to be written in MEM. And the third instruction writes the latest value of $t1 to the register files thus the value of $t1 is changed to 0x08080404.

In cycle 9, the nop is fetched. The seventh instruction enters the ID stage while the sixth instruction is in the EX stage. The fifth instruction is in MEM stage while the fourth instruction is in WB stage. However the sixth instruction needs the latest value of $t2 to write to the DMEM, so the value of data from WB stage is forwarded to the second input of ALU, which can be seen from the $DEBUG\_FORWARDB$ signal, but it is not used by the ALU because the ALU calculates the address of the data to be written in MEM. And the fourth instruction writes the latest value of $t2 to the register files thus the value of $t2 is changed to 0x04040202. The fifth instruction writes to DMEM so at the end of the cycle, the value of DMEM(4) is changed to 0x04040202.

In cycle 10, the nop is fetched. The eighth instruction enters the ID stage while the seventh instruction enters the EX stage. The sixth instruction enters MEM stage and the fifth instruction enters WB stage. The sixth instruction writes to DMEM so at the end of the cycle, the value of DMEM(4) is changed to 0x04040404 and the value of DMEM(8) is changed to 0x02020000.

In cycle 11, the nop is fetched. The sixth instruction enters WB stage while the other stages is full of nop.

2 Extra Credit

If running the test program with a nop inserted after the first instruction, then there is no need to stall the pipeline because by the time the second instruction reaches the EX stage, the value of latest $t0 would have already been in WB stage, so the value in WB stage will be forwarded to the input of ALU. The number of cycles needed to finish the program would still be the same as the test program which is 11 because in execution of the test program a nop is automatically added in the pipeline. The value of PC would be 0x18 when the final sw occurs because another instruction is inserted in the program before it. The state of DMEM/Registers would be the same as the test program.