

# REPORT - 9

-SIDDHANT CHOUHDARY  
-HARDIK AGGARWAL

The goal of the assignment was to implement the software for the pipeline model of a MIPS processor with maximum efficiency in terms of cycles. In C++. The whole process could be divided into 5 stages – Instruction fetch, Instruction Decode, ALU, Memory access, Register write. For the implementation, we have used a loop for the process and each iteration in the loop corresponds to one cycle. The instructions are fetched line by line from an input file and stored in an array of strings.

1. In the instruction fetch state, an instruction is fetched from the block memory.
2. In the next iteration it goes to the decode stage where the opcodes and register values are updated. Two control signals mux\_RegDst and mux\_Alusrc are also updated in this stage.
3. In the ALU stage, the arithmetic and logical operations take place along with updating of mux\_MemRead and mux\_MemWrite signals.
4. In the memory stage, the data is either written to the memory or taken from the memory depending upon the value of control signals. mux\_MemtoReg and mux\_MemWrite control signals are updated here.
5. In the last stage, the data is written back to the destination register. Since this happens in the first half of the cycle, the condition has also been placed near the start of the loop. At the end of the loop the value of the pc is updated.

The final values stored in each of the registers is shown in the output along with the number of cycles involved as well as the IPC.

In the last model the pipeline was slow due to several stalls. But in this model we have reduced them. The stalls due to hazard have been made zero by forwarding the values that would have been written in the register in WB for previous instructions to the EXE stage for present instructions.

The control hazard still takes one stall(bubble). So if the instruction set contains only data hazards then it would take the same number of cycles as the actual pipeline without hazards.

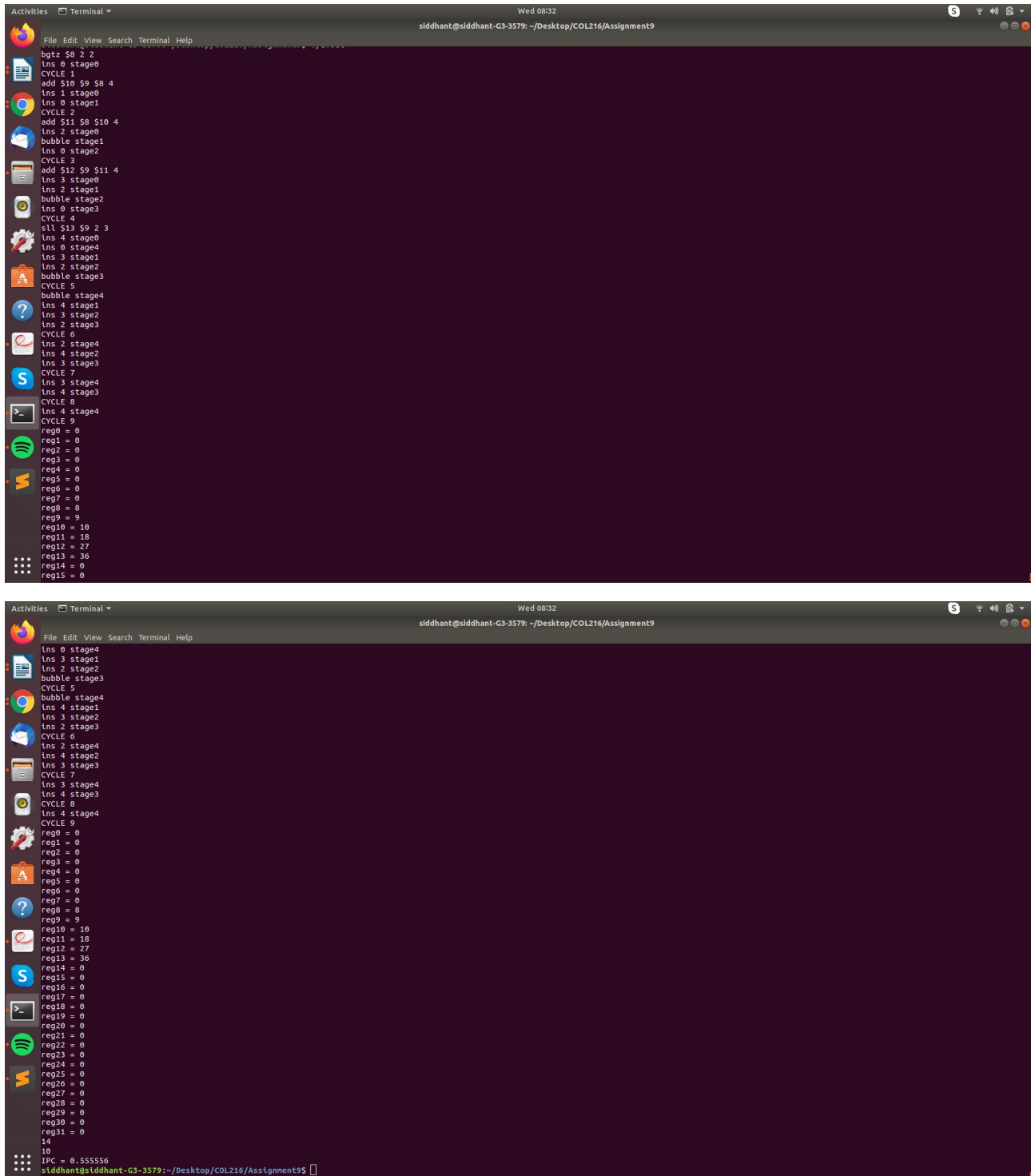
TEST CASE -

```
bgtz $8 2 2
add $10 $9 $8 4
add $11 $8 $10 4
add $12 $9 $11 4
sll $13 $9 2 3
```

INITIAL VALUES -

```
reg[8]=8;
reg[9]=9;
reg[10]=10;
ins[8]="13";
ins[9]="14";
ins[14]="15";
ins[10]="10";
```

## OUTPUT -



```
File Edit View Search Terminal Help
siddhant@siddhant-G3-3579: ~/Desktop/COL216/Assignment9

bgtr $8 2 2
ins 0 stage0
CYCLE 1
add $10 $9 $8 4
ins 1 stage0
ins 0 stage1
CYCLE 2
add $11 $8 $10 4
ins 2 stage0
bubble stage1
ins 0 stage2
CYCLE 3
add $12 $9 $11 4
ins 3 stage0
ins 2 stage1
bubble stage2
ins 0 stage3
CYCLE 4
sll $13 $9 2 3
ins 4 stage0
ins 0 stage4
ins 3 stage1
ins 2 stage2
bubble stage3
CYCLE 5
bubble stage4
ins 4 stage1
ins 3 stage2
ins 2 stage3
CYCLE 6
ins 2 stage4
ins 4 stage2
ins 3 stage3
CYCLE 7
ins 3 stage4
ins 4 stage3
CYCLE 8
ins 4 stage4
CYCLE 9
reg0 = 0
reg1 = 0
reg2 = 0
reg3 = 0
reg4 = 0
reg5 = 0
reg6 = 0
reg7 = 0
reg8 = 8
reg9 = 9
reg10 = 10
reg11 = 18
reg12 = 27
reg13 = 36
reg14 = 0
reg15 = 0

ins 0 stage4
ins 3 stage1
ins 2 stage2
bubble stage3
CYCLE 5
bubble stage4
ins 4 stage1
ins 3 stage2
ins 2 stage3
CYCLE 6
ins 2 stage4
ins 4 stage2
ins 3 stage3
CYCLE 7
ins 3 stage4
ins 4 stage3
CYCLE 8
ins 4 stage4
CYCLE 9
reg0 = 0
reg1 = 0
reg2 = 0
reg3 = 0
reg4 = 0
reg5 = 0
reg6 = 0
reg7 = 0
reg8 = 0
reg9 = 9
reg10 = 10
reg11 = 18
reg12 = 27
reg13 = 36
reg14 = 0
reg15 = 0
reg16 = 0
reg17 = 0
reg18 = 0
reg19 = 0
reg20 = 0
reg21 = 0
reg22 = 0
reg23 = 0
reg24 = 0
reg25 = 0
reg26 = 0
reg27 = 0
reg28 = 0
reg29 = 0
reg30 = 0
reg31 = 0
14
IPC = 0.555556
siddhant@siddhant-G3-3579:~/Desktop/COL216/Assignment9$
```

Here we can see on the same instruction set the pipeline is taking lesser number of cycles as compared to the last case.