ECEN 651

LAB NO. 6

Pipelined MIPS Processor

TA: Mr. Ye Wang

DATE: 11/16/2019

Student Name: Dhiraj Dinesh Kudva

UIN: 829009538

Objective: The objectives of this lab is to create a pipelined MIPS processor.

Design:

The below figure is the MIPS processor. In this case, we have considered a 5-stage pipelined MIPS architecture.

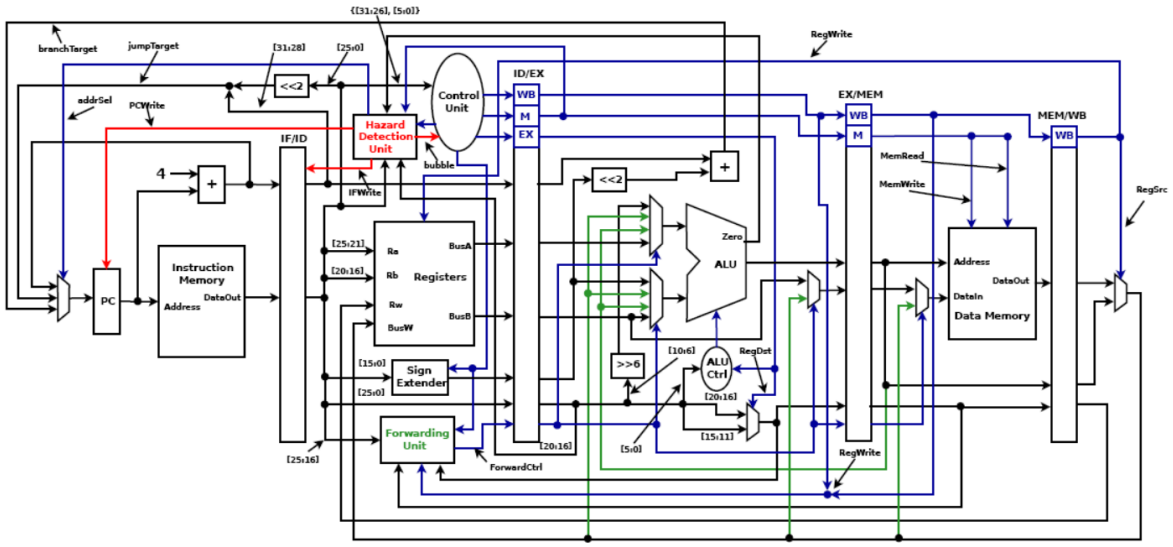


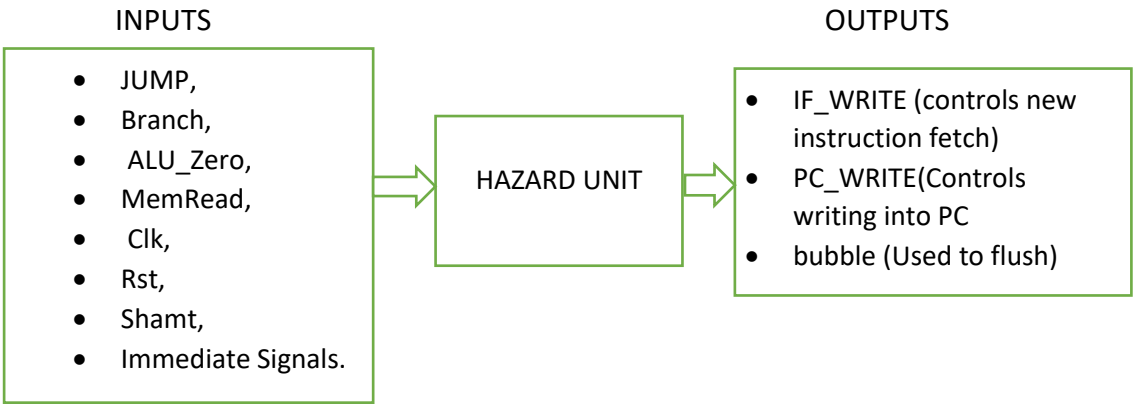
Figure 1: MIPS Block Diagram

The pipelined architecture is achieved with the help of two additional units:

1. Hazard Unit and 2. Forwarding Unit.

The main aim is to introduce bubbles or stalls and forward data when needed to ensure smooth operation.

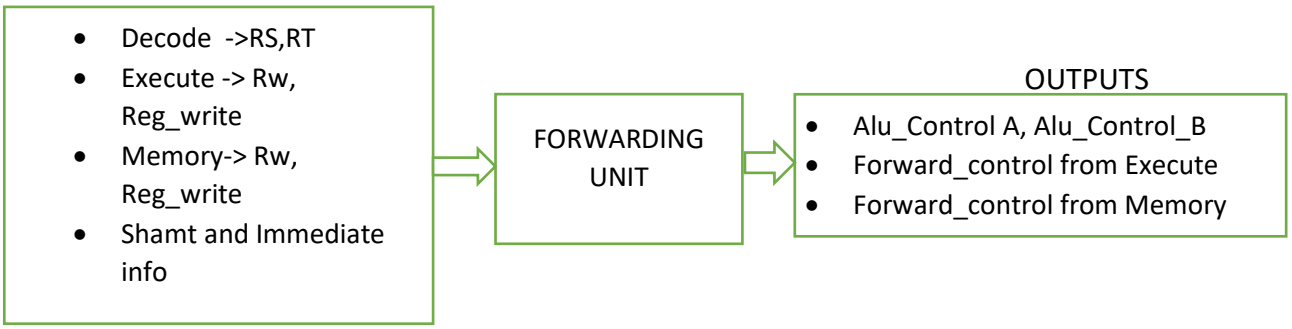
Block diagram of hazard unit is as shown below:



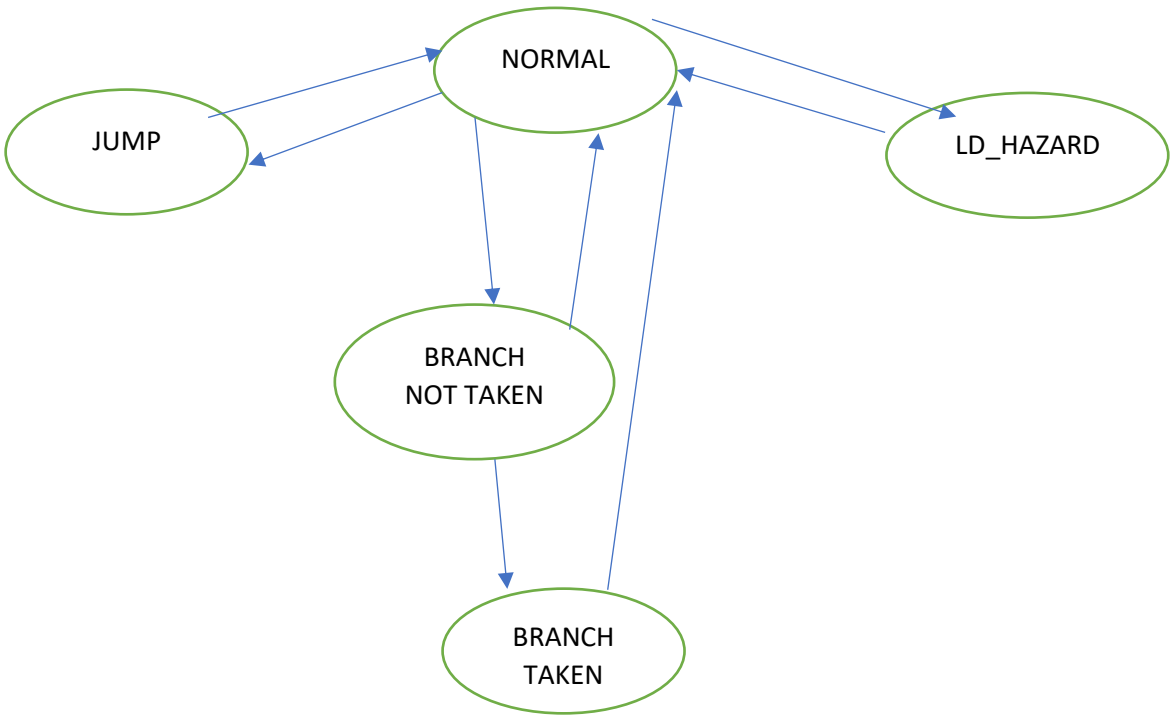
Based on the input signals, the hazard unit will generate output that will either stall the next fetch stage or allow it to proceed in the normal manner.

The forwarding unit can be logically shown as below:

INPUTS



FINITE STATE MACHINE (HAZARD UNIT)



The synthesis results and test bench results are as shown below:

Project name:	mips-pipeline
Project location:	/home/grads/d/dhirajkudva/ECEN 651/LAB7/mips-pipeline
Product family:	Virtex-7
Project part:	xc7vx485tffg1157-1
Top module name:	PipelinedProc

Synthesis		Implementation	
Status:	Complete	Status:	
Messages:	7 warnings	Messages:	N
Active run:	synth_2	Active run:	i
Part:	xc7vx485tffg1157-1	Part:	x
Strategy:	Vivado Synthesis Defaults	Strategy:	V
		Incremental compile:	N

```
# run all
             Results of Program 1 passed
             Results of Program 2 passed
            Result 1 of Program 3 passed
            Result 2 of Program 3 passed
            Result 3 of Program 3 passed
            Result 4 of Program 3 passed
            Result 5 of Program 3 passed
            Result 6 of Program 3 passed
            Result 7 of Program 3 passed
            Result 8 of Program 3 passed
            Result 9 of Program 3 passed
           Result 10 of Program 3 passed
           Result 11 of Program 3 passed
           Result 12 of Program 3 passed
All tests passed
```

Questions:

a. Instead of stalling the pipeline while waiting for a branch to be evaluated, we could simply continue to fetch from PC+4. What name is commonly given to this technique? What changes would have to be made to our existing design to accommodate this improvement? Be sure to include modifications to the Hazard Detection Unit and pipeline registers.

Ans. The technique for fetching the next instruction i.e. PC + 4 is called as static branch prediction. Static branch predictors can be used for this purpose. The primary assumption is that the branch is not taken and it proceeds with the same. This saves a lot of cycles as the cycle is not stalled, but is only useful when the primary assumption is the correct prediction i.e. the branch is not taken However if the branch is taken then we need to flush all the earlier instructions.

In order to execute this, then we need to make the following changes in the Hazard Unit:

{IF_write, PC_write, bubble, addrSel} = 5'b01110; needs to be changed to {IF_write, PC_write, bubble, addrSel} = 5'b11010;

i.e. we are not inserting the bubble in the hazard unit stage and continue to fetch the instruction. In the later two cycles, when the true outcome is known, a bubble needs to be set high. This can be done by being in the state as Branch state and comparing the true branch state and predicted branch state.

Also, the pipelined registers must monitor the bubble in the execute and memory stage as well in order to flush the calculated instruction.

b. What clock rate did the synthesis process estimate your overall design would run at? Look through the synthesis report and locate the section where the design timing is estimated. From that, determine which components are in the critical path. What changes could you make to the design to improve the clock rate without adding additional pipeline stages

Ans:

```
    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Er
    -------      -------  ---------------------  -----------
      0.074        0.000                      0

All user specified timing constraints are met.

------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk1   {0.000 3.900}        7.800           128.205
```

As shown above, the synthesis occurred at 128.205 Mhz with WNS at 0.074 ns.

```
                          (falling edge-triggered cell FDCE clocked by clk1  {rise@0.
Path Group:             clk1
Path Type:              Setup (Max at Slow Process Corner)
Requirement:            7.800ns  (clk1 fall@11.700ns - clk1 fall@3.900ns)
Data Path Delay:        7.419ns  (logic 1.106ns (14.908%)  route 6.313ns (85.092%))
Logic Levels:           10  (LUT3=1 LUT4=3 LUT5=1 LUT6=5)
Clock Path Skew:        -0.030ns (DCD - SCD + CPR)
  Destination Clock Delay (DCD):    4.909ns = ( 16.609 - 11.700 )
  Source Clock Delay      (SCD):    5.273ns = ( 9.173 - 3.900 )
```

As shown below are the delay of route is 85.092 %. Thus by making routes shorter, total delay can be reduced and thus improve the clock rate.

c. Determine the CPI for each of the three programs executing on your processor. Compare those results to the CPI of the single-cycle processor. Also, provide the average CPI of all three programs.

Ans: : Program 1: Number of instructions = 21.

Number of cycles = 46.

CPI = 46/21 = 2.19 cycles/instructions

Program 2: Number of instructions = 11.

Number of cycles = 34.

CPI = 34/11 = 3.09 cycles/instructions

Program 3: Number of instructions = 40.

Number of cycles = 38.

CPI = 40/38 = 1.05 cycles/instructions

AVG CPI(PIPELINED) = (2.19+3.09+1.05)/3 = 2.11 cycles/instructions

Similarly, AVG CPI (SINGLE CYCLE) =1.19

d. Given the clock rate and average CPI, compute the MIPS (Million Instructions per Second) rating for both versions of processor. Which one is faster and why?

Ans: Clockrate_pipelined = 128.205 Mhz
Avg CPI_pipelined = 2.11 clock cycles/instructions
Clockrate_singlecycle = 41.053 MHz
AVG CPI_single cycle = 1.19
MIPS = (clockrate/Avg CPI)/1,000,000
Therefore
MIPS_pipelined = (128.205/2.11) = 60.760MIPS
MIPS_singlecycle = 41.053/1.19 =34.498 MIPS

MIPS pipelined processor are faster as they can execute multiple instructions at a same time in parallel. Because of there is not much delay in waiting for other instructions, hence more instructions complete their executions per unit time. As a result of this pipelined processors provide higher throughput / performance.