Christopher Lall

CSc 342/343 – Professor Gertner

Lab Assignment – BEQ\_BNE\_J

Spring 2022

Due 4/24/22

Text, letter

Description automatically generated

Table of Contents

[Objective: 3](#_Toc101352849)

[Components/VHDL Code: 4](#_Toc101352850)

[2:1 Mux 4](#_Toc101352851)

[32bit Register: 5](#_Toc101352852)

[Data Memory: 6](#_Toc101352853)

[Instruction memory: 6](#_Toc101352854)

[Instruction Register: 7](#_Toc101352855)

[Register file: 7](#_Toc101352856)

[NAL Unit: 8](#_Toc101352857)

[Sign Ext: 8](#_Toc101352858)

[LPM Comparator: 9](#_Toc101352859)

[LPM Adder: 11](#_Toc101352860)

[Components Package: 12](#_Toc101352861)

[Simulation/Waveform: 14](#_Toc101352862)

[BEQ: 14](#_Toc101352863)

[BNE: 14](#_Toc101352864)

[Explanation analysis: 15](#_Toc101352865)

[BEQ: 15](#_Toc101352866)

[BNE: 17](#_Toc101352867)

# Objective:

This lab's goal is to teach me how to utilize a comparator to determine whether or not a condition has been met. Through this lab, I learned how the MIPS processor's BNE, BEQ, and J functions. To get to this answer, we had to employ many components such as a pc adder, mux, pc register, general register, instruction register, controller, and finally a test bench to determine if our design was operating properly.

# Components/VHDL Code:

## 2:1 Mux

A picture containing table

Description automatically generated

Figure 1. 2:1 mux vhdl code pt 1

Text

Description automatically generated

Figure 2. 2:1 mux vhdl code pt 2

## 32bit Register:

Text

Description automatically generated

Figure 3. 32bit register vhdl code

## Data Memory:

Text

Description automatically generated

Figure 4. data mem vhdl code

## Instruction memory:

Text

Description automatically generated

Figure 5. instruction mem vhdl code

## Instruction Register:

Text

Description automatically generated

Figure 6. instruction reg vhdl code

## Text Description automatically generated with low confidenceRegister file:

Figure 7. reg file vhdl code

## NAL Unit:

Text

Description automatically generated

Figure 8. NAL unit vhdl code

## Sign Ext:

Text

Description automatically generated

Figure 9. sign ext vhdl code

## LPM Comparator:

Text

Description automatically generated

Figure 10. LPM Comparator vhdl code pt 1

Text, letter

Description automatically generated

Figure 11. LPM Comparator vhdl code pt 2

## LPM Adder:

Text, timeline

Description automatically generated

Figure 12. LPM Adder vhdl code pt 1

Text

Description automatically generated

Figure 13. LPM Adder vhdl code pt 2.

## Components Package:

Text

Description automatically generated

Figure 14. Components package vhdl code pt 1

A picture containing text

Description automatically generated

Figure 15. Components package vhdl code pt 2

Predefined arbitrary data will be stored in the register file component at the arbitrary address indices Rs and Rt. The corresponding data will be transferred to outputs BusA and BusB based on the values of Rs and Rt (which are fed from the IR). WREN, RST, Waddress, and Data are all present as inputs, and behaviors are defined, but they will not be used in this lab. Because we aren't writing to any registers, the inputs could be eliminated if necessary.

The 32-Bit Comparator will compare the outputs of the Register File (i.e. BusA and BusB) to set a condition flag CondEqual either to a 1 or a 0.

The Program Counter is a 32-bit register with the primary purpose of updating the current PC's value. At the rising edge of the clock, the PC value will be updated.

The sign extender merely converts the 16-bit Immediate field to 32-bits, which is required to identify the following instruction's address. To retain the sign of the value, the remaining 16 bits will be filled with either 0's or F's depending on whether the most significant bit is positive (0–7) or negative (8–F).

Two adder components will be used to compute the two possible PC values as described earlier.

Another required component is a 2:1 Multiplexer which will use the CondEqual flag to select one of the two computed PC values to feed back into the PC and give the final PC value.

A package is created to store all components.

With the utilization of each component, we create a symbol so that we can realize the NAL unit

# Simulation/Waveform:

## BEQ:

A picture containing treemap chart

Description automatically generated

Figure . modelsim sim for BEQ instruction

## BNE:

Graphical user interface, diagram

Description automatically generated

Figure 17. Verification 1

Graphical user interface

Description automatically generated

Figure 18. Verification 2

A screenshot of a computer

Description automatically generated

Figure 19. Verification 3

Diagram, schematic

Description automatically generated

Figure 20. Total verification

# Explanation analysis:

## BEQ:

Three verifications will be performed to ensure the unit's accuracy. A 32-bit arbitrary instruction is supplied into the Instruction Register (IR) for each verification, which decodes the Rs and Rt address indices as well as the 16-bit Immediate. These indices are used as inputs to the register file, which uses them to store predefined arbitrary 32-bit data. After then, the 16-Bit Immediate is sign expanded. This information is sent to BusA and BusB, which are compared using a comparator and a condition flag is set (1 if equal, 0 if not). With a predefined arbitrary program counter value, the two possible addresses are computed using an adder and are fed to a 2:1 multiplexer which will select the final address of the next instruction based on the condition flag set earlier (i.e. select address 2 if condition flag is 1, else select address 1).

Verification 1: The IR was programmed with the arbitrary 32-bit instruction 0x74BB4AD3. This instruction is decoded by the IR to produce the indices Rs = 0x05 and Rt = 0x1B, as well as the 16-Bit Immediate field = 0x4AD3. 0x00004AD3 is the signed extended ImmExt. 0x10000004 is an arbitrary 32-bit PC value. For indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0x3FFD02E1 and 0x224CCC80, respectively. This information is sent to Bus A and Bus B. Due to the fact that the data stored on these buses is not equal, the condition flag is set to 0. As a result, the expected output (i.e. the next instruction's address) is PC + 4 (i.e. InstrAddr = 0x10000008).

Verification 2: We used an arbitrary 32-bit instruction 0xF2AD52C0 in the IR for the second verification. This instruction is decoded by the IR to produce the indices Rs = 0x15 and Rt = 0x0D, as well as the 16-Bit Immediate field = 0x52C0. 0x000052C0 is the signed extended ImmExt. The PC value remains the same. For indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0xFFFFEEEE and 0xFFFFEEE, respectively. This information is sent to Bus A and Bus B. Because the data stored on these buses is identical, the condition flag is set to 1. As a result, the intended output (i.e. the next instruction's location) is PC + ImmExt + 4. As a result, the NextAddr instruction's address is 0x100052C8.

Verification 3: We inserted an arbitrary 32-bit instruction 0xED2080F2 into the IR for the third verification. This instruction is decoded by the IR to produce the indices Rs = 0x09 and Rt = 0x00, as well as the 16-Bit Immediate field = 0x80F2. 0xFFFF80F2 is the signed extended ImmExt. The PC value remains the same. For indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0x00C3A23B and 0x00C3A23B, respectively. This information is sent to Bus A and Bus B. Because the data stored on these buses is identical, the condition flag is set to 1. As a result, the intended output (i.e. the next instruction's location) is PC + ImmExt + 4. As a result, the NextAddr instruction's address is 0x0FFF80FA.

## BNE:

Verification 1: The IR was programmed with the arbitrary 32-bit instruction 0x74BB4AD3. This instruction is decoded by the IR to provide address indices Rs = 0x05 and Rt = 0x1B, as well as 16-Bit Immediate field = 0x4AD3. 0x00004AD3 is the symbol extended instant (ImmExt). 0x10000004 is an arbitrary 32-bit PC value. For the indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0x3FFD02E1 and 0x224CCC80, respectively. This information is sent to Bus A and Bus B. Due to the fact that the data stored on these buses is not identical, the condition flag is set to 1. PC + ImmExt + 4 = 0x10004ADB is the address of the following instruction.

Verification 2: We used an arbitrary 32-bit instruction 0xF2AD52C0 in the IR for the second verification. This instruction is decoded by the IR to give address indices Rs = 0x15 and Rt = 0x0D, as well as 16-Bit Immediate field = 0x52C0. 0x000052C0 is the sign extended instantaneous (ImmExt). The PC value remains the same. For address indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0xFFFFEEEE and 0xFFFFEEE, respectively. This information is sent to Bus A and Bus B. Because the data on these buses is identical, the condition flag is set to 0. PC + 4 = 0x10000008 is the address of the following instruction.

Verification 3: We inserted an arbitrary 32-bit instruction 0xED2080F2 into the IR for the third verification. This instruction is decoded by the IR to provide address indices Rs = 0x09 and Rt = 0x00, as well as 16-Bit Immediate field = 0x80F2. 0xFFFF80F2 is the signed extended (ImmExt). The PC value remains the same. For address indices Rs and Rt, the arbitrary 32-bit data (stored in the register file) is 0x00C3A23B and 0x00C3A23B, respectively. This information is sent to Bus A and Bus B. Because the data on these buses is identical, the condition flag is set to 0. PC + 4 = 0x10000008 is the address of the following instruction.