

Lab 5 Report

|  |  |
| --- | --- |
| Name 1:Brandon Hoang | Name 2: |
| Course: EGCP-381 | Date: 4/10/2019 |

Grading Criteria:

|  |  |  |
| --- | --- | --- |
| **Section** | **Earned Points** | **Possible Points** |
| Problem/Objective: |  | 5 |
| Method: |  | 20 |
| Question(s): |  | 30 |
| Program Code: |  | 20 |
| Demo: |  | 25 |
| Total: | 0 | 100 |

Report Submission Instructions:

* Only **two** people per group and **both members must upload the same report.**
* Please, upload your report to TITANIum
* No paper submissions
* When showing your work, you can use MS Word’s equation tool. Or you can hand write your work, take a picture, and paste the image here. One app that I would suggest to easily do this is “CamScanner”.
* When giving the screenshots, please take a screenshot of the whole screen (i.e., include the OS taskbar, date, clock, etc.). No cropping.

# Problem/Objective

State the problem statement and/or objective of the lab. This must be a complete paragraph (i.e., at least 5 sentences).

***In this lab, we will be revisiting the Assembly language and writing code in the PBlazeIDE. The task of this lab will be to implement a BCD conversion, but representing the decimal value in binary. We will be using a given algorithm to convert the binary input into a decimal number represented by 4 bits for each ten’s place. This lab will involve mainly left shifts in its logic.***

# Methodology

You are to search on the web (i.e., [CSUF library](http://library.fullerton.edu/), [Google Scholar](http://scholar.google.com/), etc.) to find a scholarly paper (i.e., 1 paper) on an application of Binary-to-BCD conversion. Briefly describe (about 2 complete paragraphs) what the paper is about.

https://ieeexplore.ieee.org/abstract/document/5496752

**The paper basically proposes a high speed, low power architecture for binary to BCD conversions with fixed bits. They estimate about a 28% increase in performance compared to current designs. Performance is comprised of latency, area, and power consumption. The application target of this design is for Decimal Arithmetic with floating points.   
 Decimal multiplication has prior research for binary to BCD conversions. A recoding scheme used for partial product generation is applicable for signed magnitude partial products. Data independent optimization techniques have also been proposed in the past for reducing average latency of arithmetic implementation. The designers of the proposed design have essentially built upon these past ideas to iterate an improved design**

# Question(s)

Convert 2610, step-by-step, to BCD in the table below. Assume the input is represented using an 8-bit unsigned binary input. See the instructions for an example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | | Special BCD Shift Register | | Binary Input |
| BCD Digit 1 | BCD Digit 0 |
| Initial |  |  |  | 0001 1010 |
| Bit 7 | No Adjustment Shift 1 bit Left |  | 0  010 | 0011 0100 |
| Bit 6 | No Adjustment Shift 1 bit Left |  | 00  010 | 0110 1000 |
| Bit 5 | No Adjustment Shift 1 bit Left |  | 000  010 | 1101 0000 |
| Bit 4 | No Adjustment Shift 1 bit Left |  | 0001  110 | 1010 0000 |
| Bit 3 | No Adjustment Shift 1 bit Left | 0  010 | 0011  310 | 0100 0000 |
| Bit 2 | No Adjustment Shift 1 bit Left | 00  010 | 0110  610 | 1000 0000 |
| Bit 1 | BCD 0 adjustment  Shift Left 1 bit | 001  110 | 1001  910  0011  310 | 0000 0000 |
| Bit 0 | No Adjustment Shift 1 bit Left | 0010  210 | 0110  610 | 0000 0000 |

# Program Code

Copy your code here. Please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your code as text, not as a screenshot/image**.

***; PBlazeIDE Template***

***; =====================================================***

***; data constants***

***; =====================================================***

***EINT***

***zeroN EQU 0***

***Three EQU 3***

***LowWipe EQU 240***

***HighWipe EQU 15***

***Max EQU 255***

***Over EQU 100***

***UpFive EQU 80***

***; =====================================================***

***; register aliases***

***; =====================================================***

***; =====================================================***

***; port aliases***

***; =====================================================***

***; input ports***

***sw\_port DSIN $00***

***; output ports***

***led\_port DSOUT $80***

***; =====================================================***

***; main program***

***; =====================================================***

***; main***

***loop:***

***LOAD s4, 0***

***IN s0, $00***

***IN s6, $00***

***LOAD s5, zeroN***

***check:***

***ADD s4, 1***

***LOAD s1, s5***

***LOAD s2, s5***

***AND s1, HighWipe***

***AND s2, LowWipe***

***COMP s1, 5***

***JUMP NC, LowThree***

***COMP s2, UpFive***

***JUMP NC, HighThree***

***proc:***

***SL0 s0***

***JUMP C, One***

***JUMP NC, Zero***

***lastCheck:***

***COMP s4, 8***

***JUMP NZ, check***

***COMP s6, Over***

***JUMP NC, Overflow***

***JUMP cut***

***cut:***

***oi EQU s5***

***OUT oi, $80***

***JUMP loop***

***; =====================================================***

***; subroutines***

***; =====================================================***

***One:***

***SL1 s5***

***JUMP lastCheck***

***Zero:***

***SL0 s5***

***JUMP lastCheck***

***LowThree:***

***ADD s5, 3***

***JUMP proc***

***HighThree:***

***ADD s5, 30***

***JUMP proc***

***Overflow:***

***LOAD s5, Max***

***JUMP cut***

