CS 350: Project One

Project Due:

October 27, 2023, 11:55 pm

Planning Document

November 10, 2023, 11:55 pm

Report and Code

This problem solving challenge is two fold, as you must provide a technical solution that is both **ethical** and **effective**. For the purposes of this assignment, we will draw our understanding of ethics from the IEEE Code of Ethics. Please review these standards. Keep them in mind as you plan and implement your solution, as you will be required to report how your solution aligned with at least one of the IEEE standards in your final deliverable.

This project also simulates a professional approach following the engineering design process. As such, your planning document and report will be written for an imagined outside audience. For example, you might imagine you are a software engineer submitting your report to your immediate supervisor.

In this project, you will respond to a documented vulnerability in hardware/software interfaces: bitflips. As you will learn in a video from Veritasium (link on Blackboard), bitflips can be caused by interference from subatomic particles that "flip" a bit in your binary code by charging a transistor in your hardware. Basically, under the right conditions, what should have been a "0" can become a "1" (or vice versa) and create a cascade of problems. Bit flips can also occur during long term storage of data, or communication over wire/wireless/fiber/etc.
Veritasium provides three documented examples of bitflip interference:

- A speedrunner receives an unearned boost
- A plane plummets and causes injuries
- A candidate receives more votes than possible

Your objective is to create a solution to one of the situations as presented in the video. Your solution should contribute to overall system resiliency by mitigating the impact of this vulnerability in software/hardware interfaces. Specifically, your solution should allow for error detecting and error correcting codes to be used on potentially vulnerable data. Your system will use the Hamming Code technique and perform three tasks:

- Encoding the data (e.g. for storage or transmission)
- Detecting and correcting errors. (e.g. after reading the stored data, or after reception)
- Decoding the data (e.g. to retrieve the original data)

The operation performed will be determined by a user input. Your solution must be user friendly, and make it clear how to interact with it. E.g. selecting operation, input/output of data, etc.

Deliverables

This project will require you to submit the following **deliverables** on Blackboard:

- Due October 27, 2023, 11:55 pm
  - Planning document (in **PDF** format only)
- Due November 10, 2023, 11:55 pm
  - Code (MIPS assembly **.s** code, in bare mode)
  - Report (in **PDF** format only)

Constraints/Specification Requirements:

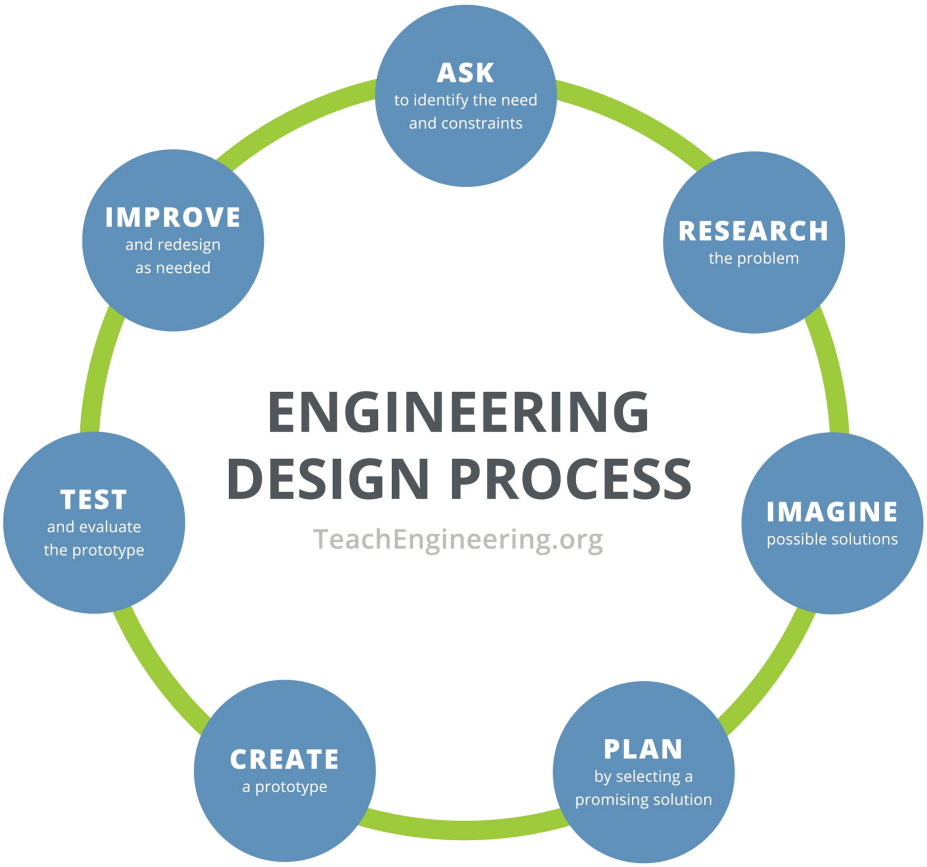
- Must be designed in the setting of an example from the video
- Proper commenting
- Must use (15+1,11) Extended Hamming Code
- No pseudo instructions, must work in bare mode (simple mode is accepted with penalty)
- Pad any data with zeros as needed
- Assume standard word size memory slots
- Multiple addresses in memory will be used
  - One buffer for the input string from user
  - One for the input data/codeword (converted from input string)
  - One for each of the 5 parity bits
  - One for the full 16-bit encoded codeword from data
  - One for the unvalidated extracted 11-bit data
  - One for combined 4-bit Hamming syndrome
  - One for Total Parity (p0) validity check (1 for False/error, 0 for True/valid)
  - One for fixed/corrected codeword
    - If there is no error, this is a duplicate of the input codeword.
  - One for final correct output data
    - If there is no error, this is a duplicate of the unvalidated extracted data
- User controlled behavior (Encode/Decode/Terminate)
  - Refer to the details

Getting started and Planning

Due October 27, 2023, 11:55 pm

This is a solutions-oriented project, and good solutions come from a strong grasp of the problem. So, begin with the “ask” and “research” steps to define the problem and constraints before imagining solutions and considering implementation.

Create a [Word/GoogleDoc] document file to contain all of your written work for this stage of the project. Name the file **CS350_project1_stage1_*username***, where ***username*** refers to your IIT username. That is, the part of your email address that comes before “@hawk.iit.edu”. Put your name in the header and right-align it



Part A: Project Setup

Choose one of the example situations from the video to be the situation for your project.
Consider the bigger picture of the main problem you are trying to solve. Without considering any details or specifications mentioned, ask below questions and add your answers to your document. Make sure they are clear and concise.

1. What “need” will your project address? What are the causes and effects of the problem?
2. What are the inputs?
3. What are the outputs?
4. What is the objective(s) of the design you will be implementing? What will it accomplish and who will benefit from the design?

When this is complete, this document should be used for the remaining parts.

Part B: Research

Hamming code is a critical method in data storage and communications. It consists of parity bits that correspond to specific data bits.

Here are some extra resources you can look at to understand how Hamming Code works:

- “How to send a self-correcting message (Hamming codes)” video by 3Blue1Brown https://www.youtube.com/watch?v=X8jsijhllIA
- Hamming Code on Wikipedia (**Flipped order of bits**) https://en.wikipedia.org/wiki/Hamming_code
- Explanation and examples, “Hamming Codes: Detecting and Correcting”, F. Gasman http://dimacs.rutgers.edu/archive/drei/1997/classroom/lessons/hamming.html
- “Fundamentals in Information Theory and Coding”, Borda, Monica (Chapter 5.7.9) https://i-share-iit.primo.exlibrisgroup.com/permalink/01CARLI_IIT/8ucv33/cdi_askewsholts_vlebooks_9783642203473

Note that some of these resources use a flipped order of bits. (so LSB is on the left and MSB is on the right).
Also the placement of the total parity bit can vary from one source to another.
For our project, refer to the constraints and our examples.

Record the relevant information you will need in your planning document.

Part C: Project Definition

The first step in the design process is to focus on the “ask” and establish what the design is supposed to do. Make a section in your document called “Project Overview”. There, write one or more paragraphs, presenting the following information:

1. The problem
2. Causes
3. Effects
4. People impacted if left unsolved
5. Brief description of how this project addresses that need

Part D: Planning Document

Once you have imagined possible solutions, draft a plan to implement your most promising idea. Create a complete guide to implement. The requirements for your project includes the following elements:

- Graphical scaffold for implementation. Use a **Flow Chart**
    - For the functions use abstraction to represent them in the overall chart. Then have a chart for each function
    - Must be produced digitally, not handwritten or hand-drawn on a tablet. LucidChart, Powerpoint and Google Slides are all options to accomplish this task.
- Draft “header” comments for each function needed
- Define elements for testing. It is very important that you define testing **before** any implementation.
    - Identify what is being tested
    - Identify what is constant for a test
    - Identify what variables need to be changed during testing
    - Create step by step instructions on what you need to do to test that function.
        - E.g. What is the expected result? How do you determine success vs failure?
    - Create a set of 3 end-to-end Test Cases for each:
        - Encoding the data
        - Decoding the data with no errors, verifying its correctness.
        - Decoding the data with 1 error, identifying the error and correcting it
        - Decoding the data with 2 errors, identifying there are two errors

Function Being Tested	Inputs	Expected Outcome
…	…	…
Sample: Encoding	User select input **XX** mode User gives value **XX**	The correct encoded data will be **XX**
…	…	…
…	…	…

The final report is your opportunity to describe your process and present your work to your imagined audience. The required form and content for the final report are summarized below. We have also prepared a template and corresponding instructions for use (below). If you have additional questions about report writing, you might check this helpful guide in OWL or visit the IIT Writing Center .

There is no required template for planning document.

Report Requirements:

Due November 10, 2023, 11:55 pm

- **Format**: Use appropriate and professional style for your report. Bold headers and subheaders. Use third-person throughout; no pronouns.
- **Citations:** Cite all sources of information, including outside images. Provide in-text and reference page citations in IEEE citation style.

Content

The required sections (which should appear as headers) for the final report are listed below and laid out in the template.

- **Introduction-** The purpose of an introduction is to show how your project is relevant to readers' goals and interests. You accomplish this with a **System Description** and **Motivations** subsections. Basically, you want to describe the system, its purpose, and your motivations, **including the connection to industry ethics**.
- **System Development-** Describe how you designed your system. Include:
    - **Design parameters**: Describe inputs, outputs, constraints, and goals.
    - **System Diagram**: Present a flowchart (to help visualize the logic of your process)
    - **Test plan**: Define how you tested your design. Your test plan should be written with the replicability of your findings in mind.
    - **Implementation:** Describe how the program was structured and implemented.
- **Analysis and Conclusion-** Analyze your results, focusing on information that is most useful to the audience. Conclude your work by summarizing they key points.

Advisements

- **Template Use:** The provided template serves as a formatting guide. Use it to your advantage. The content suggestions [which appear in brackets] should not appear in your report. The bracketed suggestions should be deleted in your final draft. Leaving them in will impair the readability of your final draft. Likewise, the words "your name" should be replaced with your own name and should not appear in the final draft.
- **Professionalism:** If you write "I give up", "I ran out of time", "I couldn't figure it out"; we reserve the right to stop reading the section at that point. You should be able to create speculations on outcomes- write plans on what would be the next steps or hypothesis; just make sure that the hypothesis is not a result.
- **Academic Honesty**: All content must be original and will be checked for plagiarism. All figures and tables should be created digitally on your own, any use of figures from an external source without proper IEEE citation will be treated as an academic honesty

violation. If citations are needed, they belong in a reference/ bibliography section and with proper references within the text. Remember, zero tolerance for academic honesty violations.

Code Requirements

Due November 10, 2023, 11:55 pm

Your code will be submitted on Blackboard in the designated location with the name **username_project1.s** where username is your IIT username (email without @hawk.iit.edu).

All assembly code produced in this project will be tested by running the program in Qtspim, in **bare mode**. Make sure you test it carefully. To be eligible for full credit you must have your code commented.
If your code does not run in bare mode, we will test it in simple mode and apply a 10% penalty.

Implementation Details

The program should provide the user with an appropriate prompt message, and allow the user to select the operation of choice (Encode/Decode/Terminate). The program should get user input and provide appropriate output. This process should repeat as long as needed, until the user selects to terminate the program, in which case the program should terminate cleanly.

User picks terminate (‘t’ or ‘T’)

An appropriate prompt message must be shown, then the program should terminate cleanly.

User picks encode (‘e’ or ‘E’)

- An appropriate prompt message to the user to enter the raw data.
- Then user should enter data:
    - a sequence of 11 characters, ‘0’s and ‘1’s, and ENTER.
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, d_4, d_3, d_2, d_1)
        - The first 0/1 is (d_{11}), and the last 0/1 is (d_1)
- The user input should be converted and stored in the designated spot in memory
- Calculate 4 Hamming parity bits and the total parity bit, store them in their designated spots in memory and output them to the user:
    - P8=0/1
    - P4=0/1
    - P2=0/1
    - P1=0/1
    - PT=0/1
- Encode the data and parity bits into a 16-bit extended Hamming codeword, store it in the designated spot in memory.
- Output to the user:
    - Appropriate prompt, and a sequence of 16 characters, ‘0’s and ‘1’s
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, p_8, d_4, d_3, d_2, p_4, d_1, p_2, p_1, p_T)
        - The first printed 0/1 is (d_{11}) and the last one is (p_T)

User picks decode (‘d’ or ‘D’)

- An appropriate prompt message to the user to enter the raw data.
- User enters an encoded codeword. This may be a codeword with possible error(s).
    - a sequence of 16 characters, ‘0’s and ‘1’s, and ENTER.
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, p_8, d_4, d_3, d_2, p_4, d_1, p_2, p_1, p_T)'
        - The first 0/1 is (d_{11}), and the last 0/1 is (p_T)
- The user input should be converted and stored in the designated spot in memory
- Extract data bits from the received word, without any validation. Store the result data in the designated spot in memory.
- Output the unvalidated data to the user:
    - Appropriate prompt, and a sequence of 11 characters, ‘0’s and ‘1’s
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, d_4, d_3, d_2, d_1)
        - The first printed 0/1 is (d_{11}) and the last 0/1 is (d_1)
- Calculate 4 Hamming parity bits and the total parity bit, store them in their designated spots in memory and output them to the user:
    - P8=0/1
    - P4=0/1
    - P2=0/1
    - P1=0/1
    - PT=0/1
- Calculate the Hamming syndrome, store it in the designated spot in memory.
- Output the Hamming syndrome to the user:
    - Appropriate prompt, and a sequence of 4 characters, ‘0’s and ‘1’s
    - (p_8, p_4, p_2, p_1)
        - The first printed 0/1 is (p_8) and the last 0/1 is (p_1)
- Validate the overall parity bit and output to the user:
    - “Total parity - PASS”
    - “Total parity - FAIL”
- Based on the previous two steps, output the ECC test result to the user:
    - “The encoded codeword is valid”
    - “The encoded codeword has 1 error”
    - “The encoded codeword has 2 errors”

Note:
It can be assumed that there are no more than 2 errors. Your program should determine among 0/1/2 errors. This can be done using the Hamming syndrome and checking the overall parity bit. Refer to the provided sample.
The behavior of the program would depend on the result of this step.

In case of 0 error

- No correction is needed. Duplicate the codeword and the extracted data into their designated spot in memory.
- Output an appropriate prompt message.

In case of 1 error

- The error can be fixed according to the Hamming syndrome. Fix the received codeword, store it in the designated spot in memory.
- Extract the data again, this time from the corrected codeword, and store it in the designated spot in memory.
- Output the corrected codeword to the user:
    - Appropriate prompt, and a sequence of 16 characters, ‘0’s and ‘1’s
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, p_8, d_4, d_3, d_2, p_4, d_1, p_2, p_1, p_T)
        - The first printed 0/1 is (d_{11}) and the last one is (p_T)
- Output the data to the user:
    - Appropriate prompt, and a sequence of 11 characters, ‘0’s and ‘1’s
    - (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, d_4, d_3, d_2, d_1)
        - The first printed 0/1 is (d_{11}) and the last 0/1 is (d_1)

In case of 2 errors

- The codeword cannot be corrected. store -1 (negative one) in both final codeword and final data
    - -1 = 0b1111 1111 1111 1111 1111 1111 1111 1111 = 0xFFFF
- Output an appropriate prompt message.

Rubric

	Unacceptable	Nearly Meets Expectations	Complete
Planning Doc	Missing or major issues [0 Points]	Most of the sections are present, with appropriate content [10 Points]	All sections are present and the content is appropriate [20 Points]
Final Report	Missing or major issues [0 Points]	Most of the sections are present, with appropriate content [10 Points]	All sections are present and the content is appropriate [20 Points]
Code Interface	Missing, code not running [0 Points]	Code runs, interacts with user, prompts are missing or unclear, the program doesn't cover all inputs [5 Points]	Code runs, interactions are clear, the user can go through all possible inputs [10 Points]
Code Encode	Does not run or does not encode [0 Points]	Encodes but not all bits are correct [10 Points]	Encodes completely [20 Points]
Code Decode 0 errors	Does not run or does not decode [0 Points]	Decodes but cannot verify correctness [10 Points]	Decodes and verifies correctness. [20 Points]
Code Decode 1 error	Does not run or does not encode [0 Points]	Decodes but cannot detect or correct. [10 Points]	Detects and corrects the error [20 Points]
Code Decode 2 errorsw	Does not run or does not encode [0 Points]	Decodes but cannot confirm 2 errors [5 Points]	Decodes and detects 2 errors [10 Points]