

COLLEGE OF ENGINEERING

Department of Computer Engineering and Computer Science

System on Chip Specification

**Full UART & TramelBlaze**

CECS 460: System on Chip Design

Nguyen Doan

Date: 5/5/2020

In submitting this file for class work at CSULB I am confirming that this is my work

and the work of no one else. In submitting this document, I acknowledge that plagiarism in student project work is subject to dismissal from the class.

Created by Nguyen Doan on 5/5/2020

Copyright © Nguyen Doan. All rights reserved.

Table of Contents

[1. Introduction 6](#_Toc39747358)

[1.1 Purpose 6](#_Toc39747359)

[1.2 Scope 6](#_Toc39747360)

[2. Documents 6](#_Toc39747361)

[2.1 External Documents 6](#_Toc39747362)

[2.1.1 Xilinx PicoBlaze 8-bit Embedded Microcontroller User Guide 6](#_Toc39747363)

[2.1.2 Nexys 4 DDR Reference Manual 6](#_Toc39747364)

[2.2 Internal Documents 6](#_Toc39747365)

[2.2.1 Programming the TramelBlaze Using Assembly Code 6](#_Toc39747366)

[2.2.2 TramelBlaze Instructions 6](#_Toc39747367)

[3. Requirements 6](#_Toc39747368)

[3.1.1 Performance Requirements 6](#_Toc39747369)

[3.1.2 Interface Requirements 6](#_Toc39747370)

[4. Top Level Design 7](#_Toc39747371)

[4.1 Description 7](#_Toc39747372)

[4.2 Block Diagram 7](#_Toc39747373)

[4.3 Data Flow Description 8](#_Toc39747374)

[4.4 I/O 8](#_Toc39747375)

[4.4.1 Signal Names 8](#_Toc39747376)

[4.4.2 Pin Assignments 8](#_Toc39747377)

[5.1.1 Electrical Characteristics 10](#_Toc39747378)

[5.2 Clocks 10](#_Toc39747379)

[5.3 Resets 10](#_Toc39747380)

[5.4 Software 10](#_Toc39747381)

[5.5 Source Code of Top Level Design 10](#_Toc39747382)

[6. Externally Developed Blocks 11](#_Toc39747383)

[6.1 TramelBlaze 16-bit Softcore Processor 11](#_Toc39747384)

[6.1.1 Description 11](#_Toc39747385)

[6.1.2 Block Diagram 11](#_Toc39747386)

[6.1.3 I/O of TramelBlaze 12](#_Toc39747387)

[6.1.4 Register Map 13](#_Toc39747388)

[7. Internally Developed Blocks 14](#_Toc39747389)

[7.1 Full UART 14](#_Toc39747390)

[7.1.1 Description 14](#_Toc39747391)

[7.1.2 Block Diagram 14](#_Toc39747392)

[7.1.3 I/O of Full UART 15](#_Toc39747393)

[7.1.4 Register Map 15](#_Toc39747394)

[7.1.5 Verification 15](#_Toc39747395)

[7.1.6 Source Code of Full UART 15](#_Toc39747396)

[7.2 Transmit Engine 16](#_Toc39747397)

[7.2.1 Description 16](#_Toc39747398)

[7.2.2 Block Diagram 16](#_Toc39747399)

[7.2.3 I/O of Transmit Engine 16](#_Toc39747400)

[7.2.4 Verification 17](#_Toc39747401)

[7.2.5 Source Code of Transmit Engine 17](#_Toc39747402)

[7.3 Receive Engine 18](#_Toc39747403)

[7.3.1 Description 18](#_Toc39747404)

[7.3.2 Block Diagram 18](#_Toc39747405)

[7.3.3 I/O of Receive Engine 19](#_Toc39747406)

[7.3.4 State Machines 20](#_Toc39747407)

[7.3.5 Register Map 20](#_Toc39747408)

[7.3.6 Verification 21](#_Toc39747409)

[7.3.7 Source Code of Receive Engine 24](#_Toc39747410)

[7.4 Baud Decoder 25](#_Toc39747411)

[7.4.1 Description 25](#_Toc39747412)

[7.4.2 Block Diagram 25](#_Toc39747413)

[7.4.3 I/O of Baud Decode 25](#_Toc39747414)

[7.4.4 Baud Table 26](#_Toc39747415)

[7.4.5 Source Code of Baud Table 26](#_Toc39747416)

[7.5 Address Decoder 27](#_Toc39747417)

[7.5.1 Description 27](#_Toc39747418)

[7.5.2 Block Diagram 27](#_Toc39747419)

[7.5.3 I/O of Address Decoder 27](#_Toc39747420)

[7.5.4 Source code of Address Code 28](#_Toc39747421)

[7.6 PED 28](#_Toc39747422)

[7.6.1 Description 28](#_Toc39747423)

[7.6.2 Block Diagram 28](#_Toc39747424)

[7.6.3 I/O of PED 28](#_Toc39747425)

[7.6.4 Source Code of PED 28](#_Toc39747426)

[7.7 AISO 29](#_Toc39747427)

[7.7.1 Description 29](#_Toc39747428)

[7.7.2 Block Diagram 29](#_Toc39747429)

[7.7.3 I/O of AISO 29](#_Toc39747430)

[7.7.4 Source Code of AISO 29](#_Toc39747431)

[8. Chip Level Verification 30](#_Toc39747432)

[9. Chip Level Test 30](#_Toc39747433)

[10. Appendix 34](#_Toc39747434)

[10.1 Appendix 1: Assembly Source Code 35](#_Toc39747435)

[10.2 Appendix 2: Top Level Design 46](#_Toc39747436)

[10.3 Appendix 3: UART 48](#_Toc39747437)

[10.4 Appendix 4: TX\_Engine 49](#_Toc39747438)

[10.5 Appendix 5: RX\_Engine 52](#_Toc39747439)

[10.6 Appendix 6: Address Decode 56](#_Toc39747440)

[10.7 Appendix 7: Baud Decode 57](#_Toc39747441)

[10.8 Appendix 8: PED 58](#_Toc39747442)

[10.9 Appendix 9: AISO 59](#_Toc39747443)

[10.10 Appendix 10: SRflop 60](#_Toc39747444)

# Introduction

This Chip Specification documentation is intended to provide the details of the Full Universal Asynchronous Receiver and Transmitter (UART) and the interface between the UART and the 16-bit softcore processor TramelBlaze which emulates the PicoBlaze from Xilinx

## Purpose

This document will first elaborate on all the related documents and the requirements for the design. Secondly, this document will elaborate the top-level design along with the block diagram, data flow, I/O ports, Clocks, Resets, and Software. Thirdly, this document will elaborate on the details of each developed block both internally and externally. Lastly, the document will convey the Chip Level Verification and Chip Level Test to ensure the functionalities complying the requirements.

## Scope

This design is to implement the UART function fir the SOC development based on TramelBlaze microcontroller. The development can be extended in varieties of direction to provide the SOC many more interfaces or control abilities.

# Documents

## External Documents

### Xilinx PicoBlaze 8-bit Embedded Microcontroller User Guide

### Nexys 4 DDR Reference Manual

## Internal Documents

### Programming the TramelBlaze Using Assembly Code

### TramelBlaze Instructions

# Requirements

### Performance Requirements

* The UART is required to conduct the communication between the TramelBlaze microcontroller and the PC

### Interface Requirements

* Displaying a banner when starting out of reset then display a prompt
* A <CR> should send the cursor to the start of the next line and should refresh the prompt
* A <BS> should erase the character in front of the cursor - never delete the prompt
* An “\*” will result in the outputting of hometown followed by a newline and a prompt
* A “@” will result in the outputting of the number of characters received since reset followed by a newline and a prompt
* All other characters should be echoed on the display up to the 40th character – when the 40h character has been echoed then newline and a prompt should be issued

# Top Level Design

## Description

The Top Level Design contains the main blocks which represent the connections between the TramelBlaze and the Full UART.

## Block Diagram

## Data Flow Description

In order for the TramelBlaze to receive and transmit data, the raw data will first need to go through the Full UART module. The Full UART gets the configurable data from the switches then process the serial RX line by the RX\_engine and process the serial TX line by the TX\_engine.

## I/O

### Signal Names

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CLK | Input | Driven by 100MHz system clock |
| RST | Input | Active high asynchronous reset |
| switches[15:0] | Input | Input data from switches to configure UART |
| RX | Input | Serial receive data |
| TX | Output | Serial transmit data |
| LED[15:0] | Output | LED driven signals |

### Pin Assignments

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inputs**   |  |  | | --- | --- | | Signal | Pin | | clk | E3 | | rst | M18 | | switches[0] | J15 | | switches[1] | L16 | | switches[2] | M13 | | switches[3] | R15 | | switches[4] | R17 | | switches[5] | T18 | | switches[6] | U18 | | switches[7] | R13 | | switches[8] | T8 | | switches[9] | U8 | | switches[10] | R16 | | switches[11] | T13 | | switches[12] | H6 | | switches[13] | U12 | | switches[14] | U11 | | switches[15] | V10 | | **Outputs**   |  |  | | --- | --- | | Signal | Pin | | TX | D4 | | RX | C4 | | LED[0] | H17 | | LED [1] | K15 | | LED [2] | J13 | | LED [3] | N14 | | LED [4] | R18 | | LED [5] | V17 | | LED [6] | U17 | | LED [7] | U16 | | LED [8] | V16 | | LED [9] | T15 | | LED [10] | U14 | | LED [11] | T16 | | LED [12] | V15 | | LED [13] | V14 | | LED [14] | V12 | | LED [15] | V11 | |

### Electrical Characteristics

* Switches
  + 1.8V is considered Logical 1
  + 0V is considered Logical 0
* Buttons
  + 3.3V is considered Logical 1
  + 0V is consider Logical 0

## Clocks

The clock is used for the project is the 100MHz clock (pin E3) onboard the Nexys 4 DDR FPGA board

## Resets

The reset button is located at the “up button” (pin M18)

## Software

Software is written in Verilog using the Vivado 2019.2. The logic for the TramelBlaze and Full UART are written in Verilog except the Stack RAM, Scratch RAM and Instruction ROM created by IP Catalog. The .coe file is assembled from the .tba assembly file.

Appendix 1: Assembly Source Code

## Source Code of Top Level Design

Appendix 2: Top Level Design

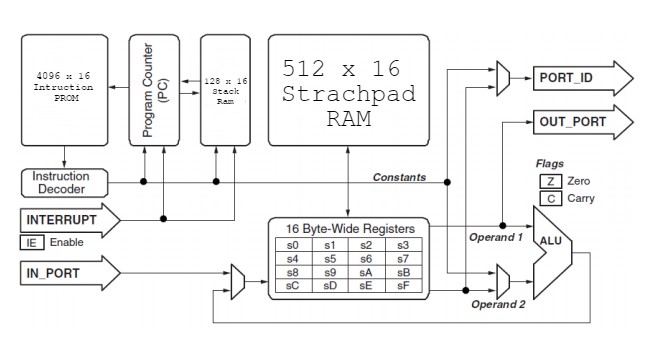
# Externally Developed Blocks

## TramelBlaze 16-bit Softcore Processor

### Description

The TramelBlaze is a 16-bit microcontroller emulating the 8-bit RISC microcontroller named PicoBlaze from Xilinx. The TramelBlaze provides cost-efficient microcontroller-based control and simple data processing. Its core is totally embedded within the target FPGA and requires no external resources.

### Block Diagram

****

### I/O of TramelBlaze

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CLK | Input | Driven by 100MHz system clock |
| RESET | Input | Active high synchronous reset |
| INTERRUPT | Input | Active high interrupt signal. When this signal is triggered, the Tramelblaze will execute the Interrupt Service Routine (ISR) |
| IN\_PORT[15:0] | Input | Data bus for inputting |
| OUT\_PORT[15:0] | Output | Data bus for outputting |
| PORT\_ID[15:0] | Output | Contains the port address for which port is inputted or outputted |
| READ\_STROBE | Output | Active high while TramelBlaze is reading |
| WRITE\_STROBE | Output | Active high while TramelBlaze is writing |
| INTERRUPT\_ACK | Output | Active high when TramelBlaze trigger after finishing the ISR |

### Register Map

|  |  |  |
| --- | --- | --- |
| **Register Name** | **Purpose** | **Description** |
| R0-RF | Holds data for the operation of the processor | This is the register files of the processor. It contains sixteen 16-bit register |
| Z | Flags if value is zero | One-bit flag |
| C | Flags if there is a carry | One-bit flag |
| IE | Flags if interrupt is enabled | One-bit flag |

# Internally Developed Blocks

## Full UART

### Description

Full UART module is designed to help the TramelBlaze communicate with other device based on UART protocol. It has the TX engine and RX engine along with supplemental logic.

### Block Diagram

### I/O of Full UART

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| WRITES[6] | Input | Active high write load. When this signal is asserted, the configuration bits will be loaded into RX engine and TX engine |
| UDI[7:0] | Input | This port takes the first 8 bits from the OUT\_PORT port of the TramelBlaze |
| RX | Input | Serial receive data |
| WRITE[0] | Input | Active high transmit trigger. This signal indicates that transmit data is ready to send |
| READ[0] | Input | Active high. When this signal is high, UART will send out the received data to the TramelBlaze |
| READ[1] | Input | Active high. When this signal is high, UART will send out the status data to the TramelBlaze |
| UDO[7:0] | Output | Data bus from UART to TramelBlaze |
| UART\_INT | Output | UART interrupt signal |
| TX | Output | Serial transmit data |

### Register Map

### Verification

### Source Code of Full UART

Appendix 3: UART

## Transmit Engine

### Description

Transmit Engine is one of two functions of the Full UART module. Transmit engine contains two big blocks and some supplemental logics. The first main block is the shift register which is utilized to shift each bit to the TX line. The second main block is the two counters; one counts for bit time and one counts for the number of transmitted bits.

### Block Diagram

### I/O of Transmit Engine

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CLK | Input | System clock |
| RST | Input | Active high synchronous reset |
| WRITE | Input | Triggers the Transmit engine to start sending data |
| EIGHT | Input | Configures the 8-bit mode |
| OHEL | Input | Configures the odd parity or even parity mode |
| PEN | Input | Configures the parity bit or not |
| K | Input | Baud rate |
| TX\_RDY | Output | Signals that the Transmit Engine is ready to send the next data |
| TX | Output | Serial transmit data |

### A close up of a green screen Description automatically generatedVerification

This is the verification for Tx Engine. The configuration is eight-bit mode and even parity. The byte 0xf3 is sent.

## Source Code of Transmit Engine

Appendix 4: TX\_Engine

## Receive Engine

### Description

Receive Engine is one of the two function of the UART module. The goal of the receive engine is to synchronize the data collection with the TX transmission from the terminal. The Receive Engine is divided into two sections, Data Path and Control Unit. The Control Unit contains two counters which counts for the bit time and the number of bits received and a state machine. The Data Path is designed to generate the received byte and the status bits such as PERR, FERR, and OVF.

### Block Diagram

### I/O of Receive Engine

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CLK | Input | System clock |
| RST | Input | Active high synchronous reset |
| READ | Input | This signal resets all the status value and the RX\_RDY signal, which indicates that the engine is ready to receive a new byte |
| RX | Input | Serial receive data |
| EIGHT | Input | Configures the 8-bit mode |
| OHEL | Input | Configures the odd or even parity |
| PEN | Input | Configurates the parity bit or not |
| K | Input | Baud rate |
| RX\_RDY | Output | Signals that Receive Engine is ready |
| UART\_RDATA[7:0] | Output | Received Data |

### State Machines

A close up of a map

Description automatically generated

### Register Map

### Verification

A screenshot of a computer

Description automatically generated

This is the verification for Rx\_engine. The configuration is eight-bit mode without parity. The received data is 0xd5. Every signal behaves properly as expected.

* PERR

A screen shot of a computer

Description automatically generatedConfiguration: 8-bit mode, odd parity

Observation: Parity is supposed be set LOW 🡪 PERR set

* A screenshot of a computer

  Description automatically generatedFERR

Configuration: 8-bit mode, no parity

Observation: STOP bit is supposed to be HIGH 🡪 FERR set

* A screenshot of a computer

  Description automatically generatedOVF
* Configuration: 8-bit mode, no parity
* Observation: When RX\_DRY and DONE are both HIGH 🡪 OVF set

### Source Code of Receive Engine

Appendix 5: RX\_Engine

## Baud Decoder

### Description

Baud decoder takes 4-bit input then decode it to the number of required clocks to obtain the intended baud rate

### Block Diagram

### I/O of Baud Decode

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CLK | Input | Driven by 100MHz system clock |
| RST | Input | Active high synchronous reset |
| Baud\_control[3:0] | Input | Input taken from switches to choose the baud rate value |
| Baud\_rate[18:0] | Output | Decoded Baud rate |

### Baud Table

|  |  |  |
| --- | --- | --- |
| Baud\_control[3:0] | Baud\_rate | Nexys 4 Count |
| 0000 | 300 | 333,333 |
| 0001 | 1200 | 83,333 |
| 0010 | 2400 | 41,667 |
| 0011 | 4800 | 20,833 |
| 0100 | 9600 | 10,417 |
| 0101 | 19200 | 5,208 |
| 0110 | 38400 | 2,604 |
| 0111 | 57600 | 1,736 |
| 1000 | 115200 | 868 |
| 1001 | 230400 | 434 |
| 1010 | 460800 | 217 |
| 1011 | 921600 | 109 |

### Source Code of Baud Table

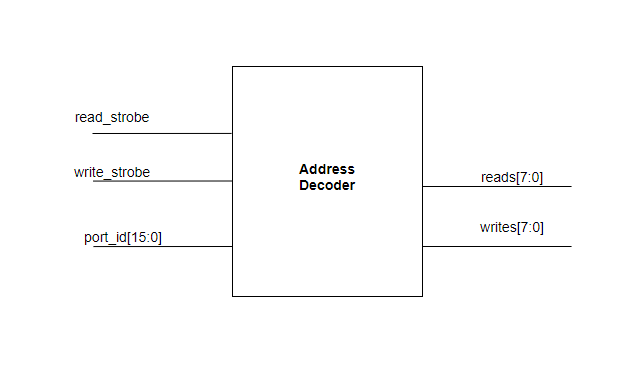
Appendix 7: Baud Decode

## Address Decoder

### Description

Address Decoder is designed to help the TramelBlaze access data from various sources within the SOC. Port\_ID[15] will decide either TramelBlaze inputs/outputs data or accesses the ROM.

### Block Diagram



### I/O of Address Decoder

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Read\_strobe | Input | Read strobe signal from TB |
| Write\_strobe | Input | Write strobe signal from TB |
| Port\_id[15:0] | Input | Port ID value from TB |
| reads[7:0] | Output | Decoded read value |
| Writes[7:0] | Output | Decode write value |

### Source code of Address Code

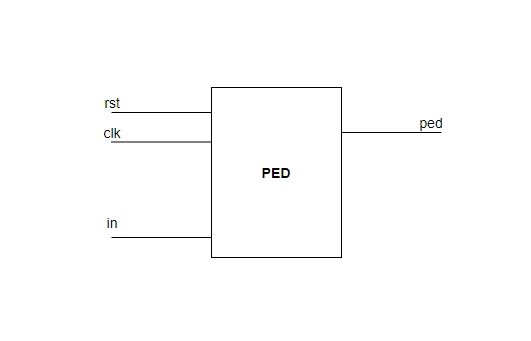
Appendix 6: Address Decode

## PED

### Description

PED is used to detect a positive edge. The output of PED will be HIGH for only one clock when a positive edge is detected

### Block Diagram



### I/O of PED

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| rst | Input | High active synchronous reset |
| clk | Input | System clock |
| in | Input | Signal input to be edge-detected |
| ped | Output | Output signal when detecting a positive edge |

### Source Code of PED

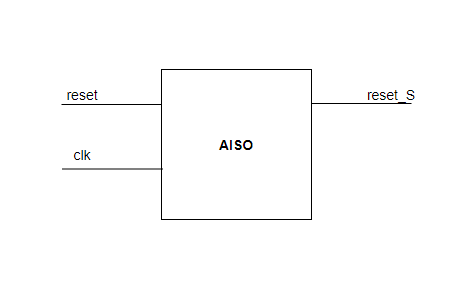
Appendix 8: PED

## AISO

### Description

AISO is utilized to synchronize an asynchronous reset. It ensures that there is no flop running into metastable state when reset is asserted

### Block Diagram



### I/O of AISO

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| reset | Input | High active asynchronous reset |
| clk | Input | System clock |
| reset\_S | Input | Synchronous reset |

### Source Code of AISO

Appendix 9: AISO

# Chip Level Verification

This is the verification for entire SOC design. After the reset is released, TX and RX begin to work.

# Chip Level Test

A screenshot of a cell phone

Description automatically generatedStarting out of reset

A screenshot of a cell phone

Description automatically generatedInput <CR>

A screenshot of a cell phone

Description automatically generatedInput “\*” to get Hometown

A screenshot of a cell phone

Description automatically generatedInput <BS>

A screenshot of a cell phone

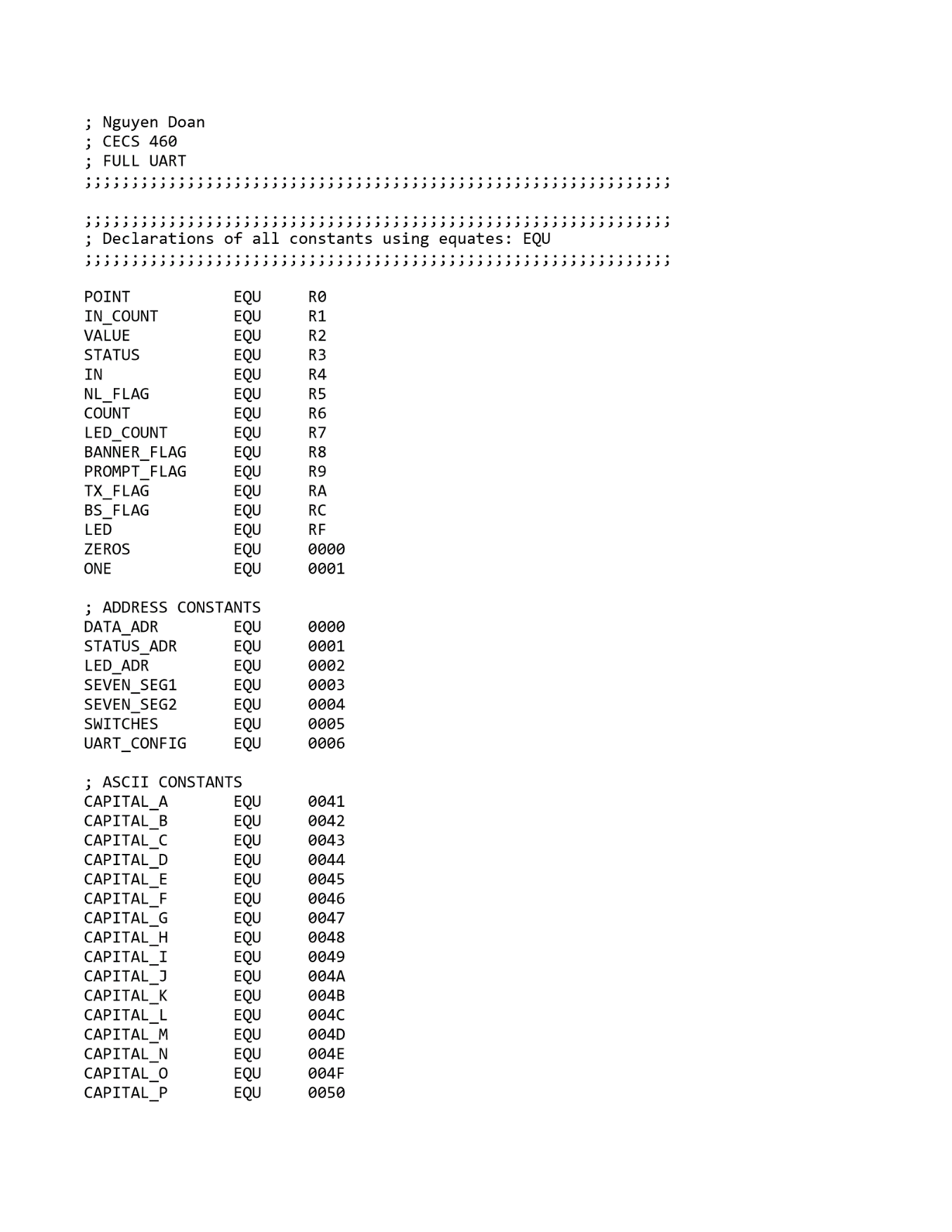
Description automatically generatedInput numbers of letter “a” to check the echoing function and new line, new prompt when the 40th character echoed

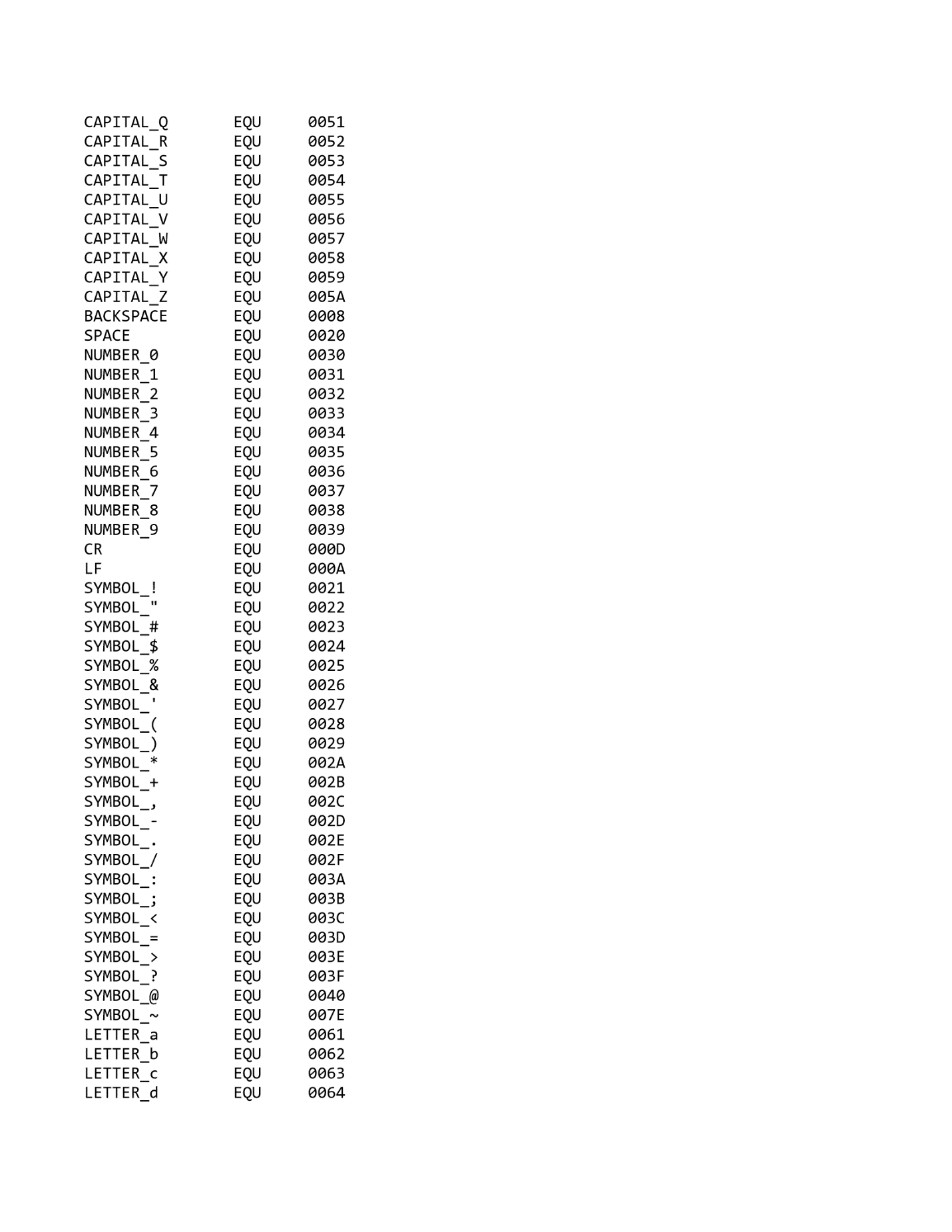
A screenshot of a cell phone

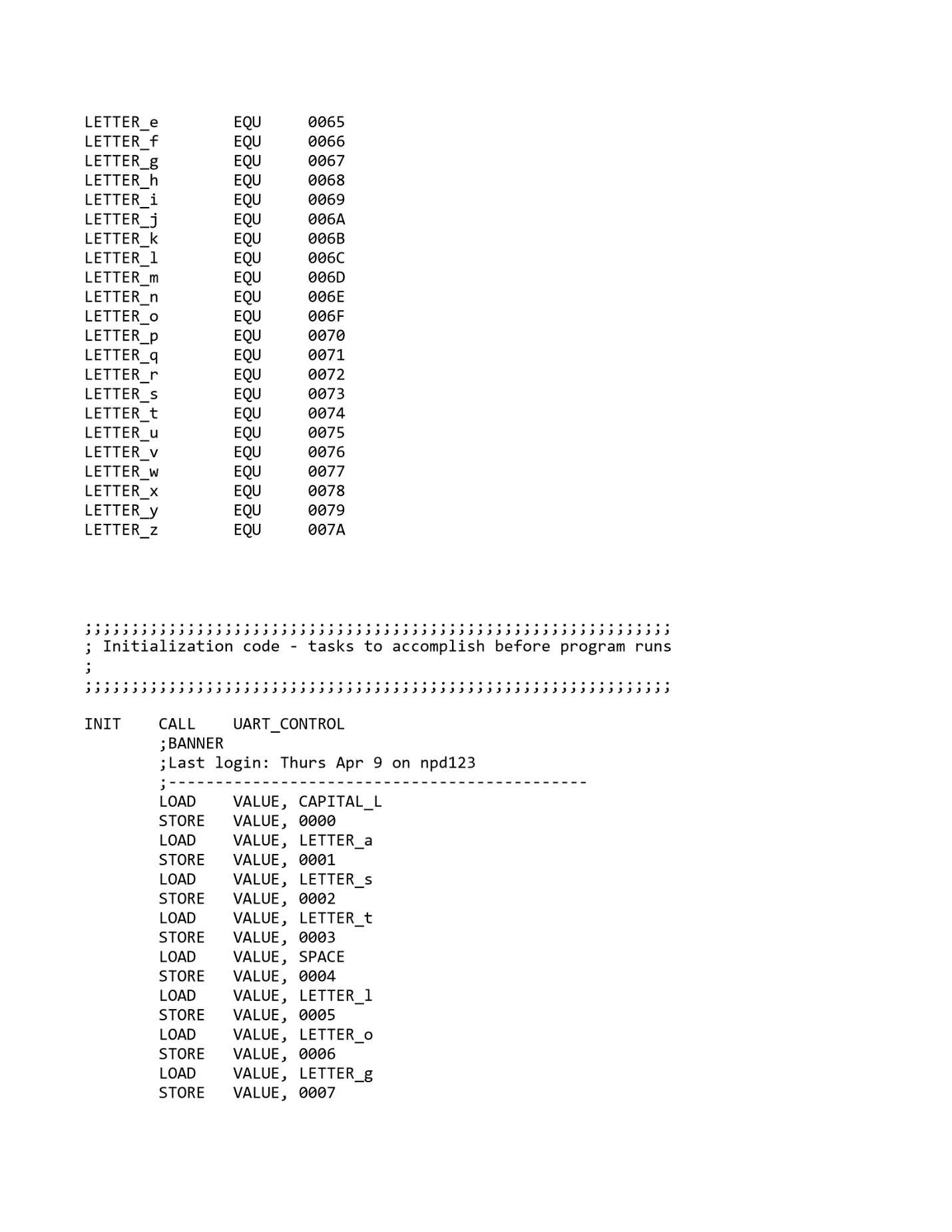
Description automatically generatedInput “@” to get the number of characters receives since reset

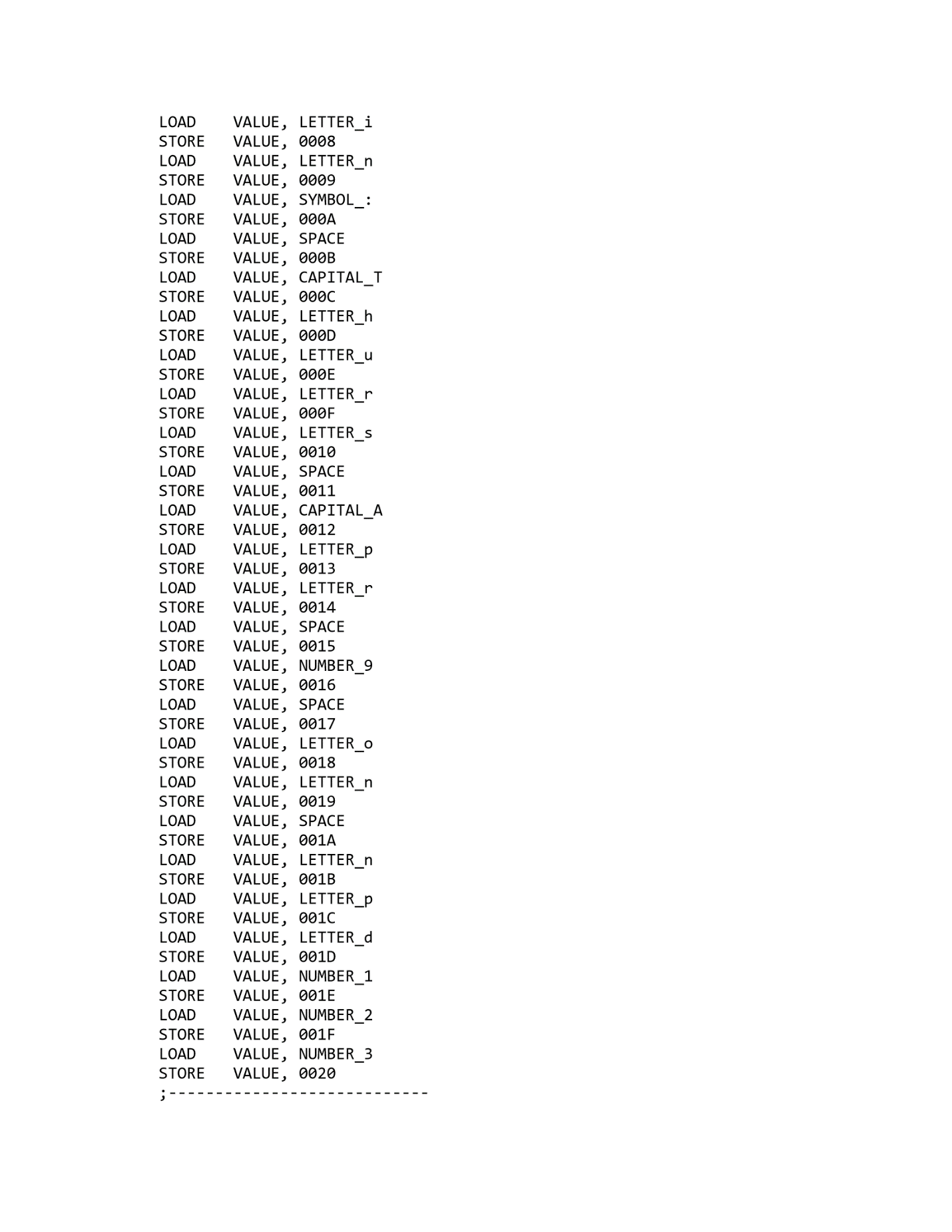
# Appendix

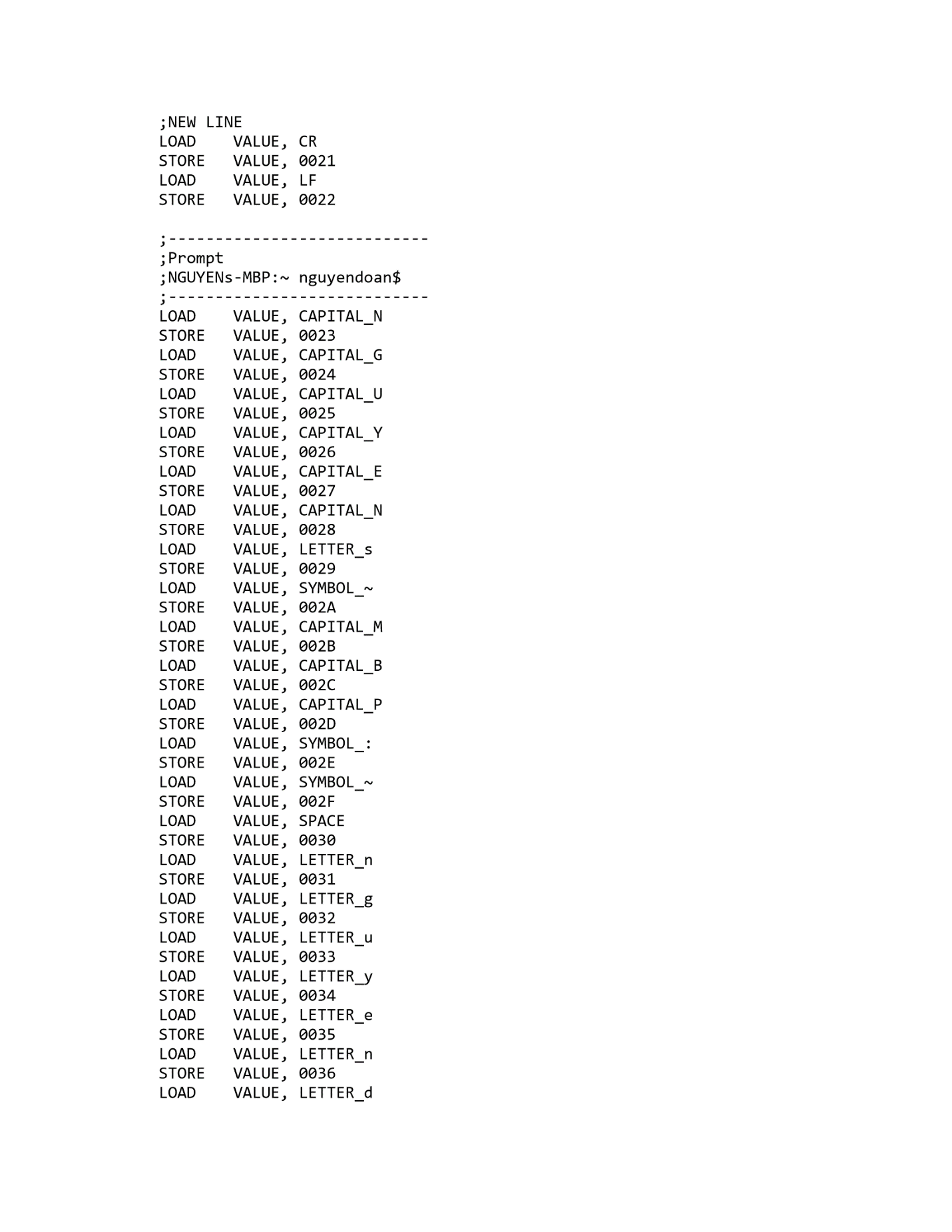
## Appendix 1: Assembly Source Code

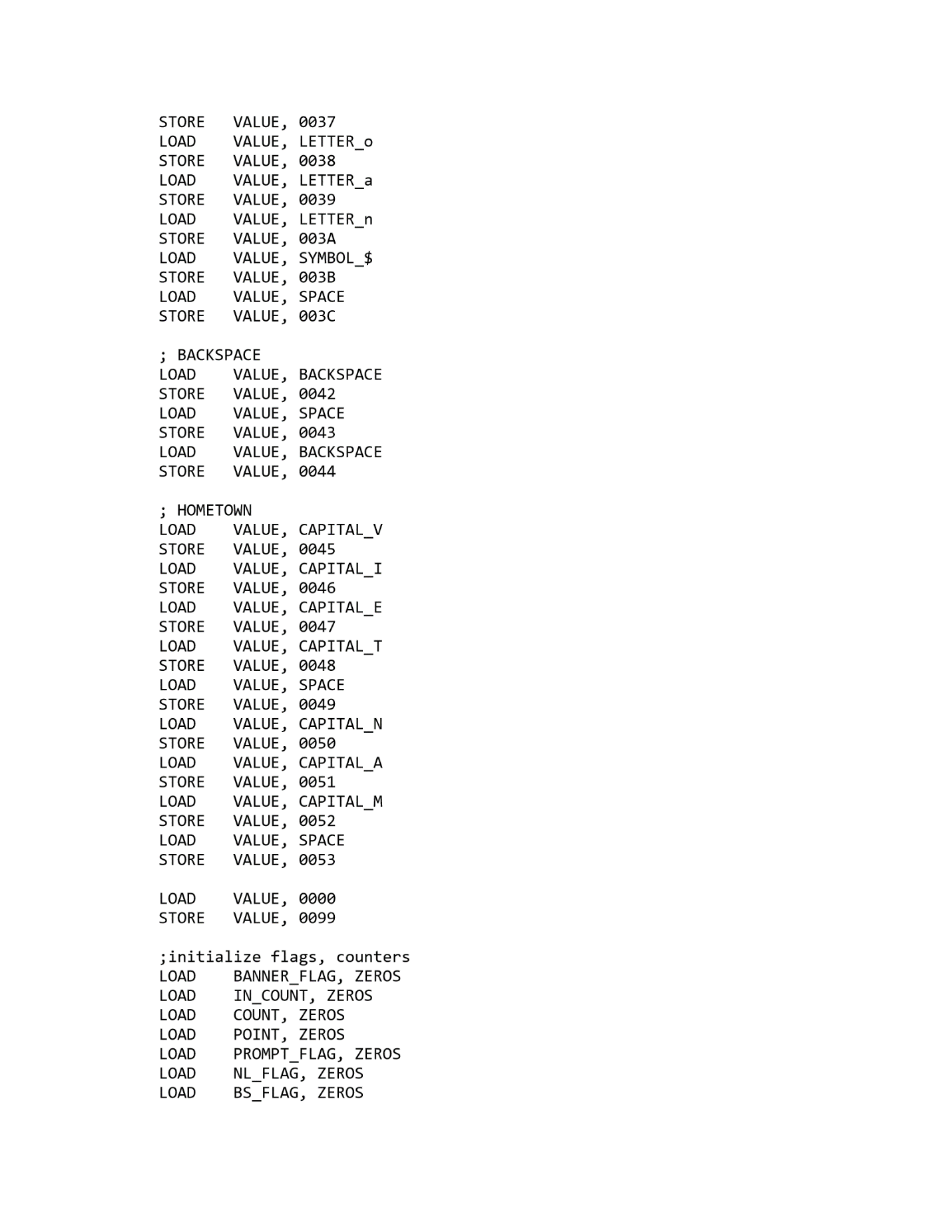


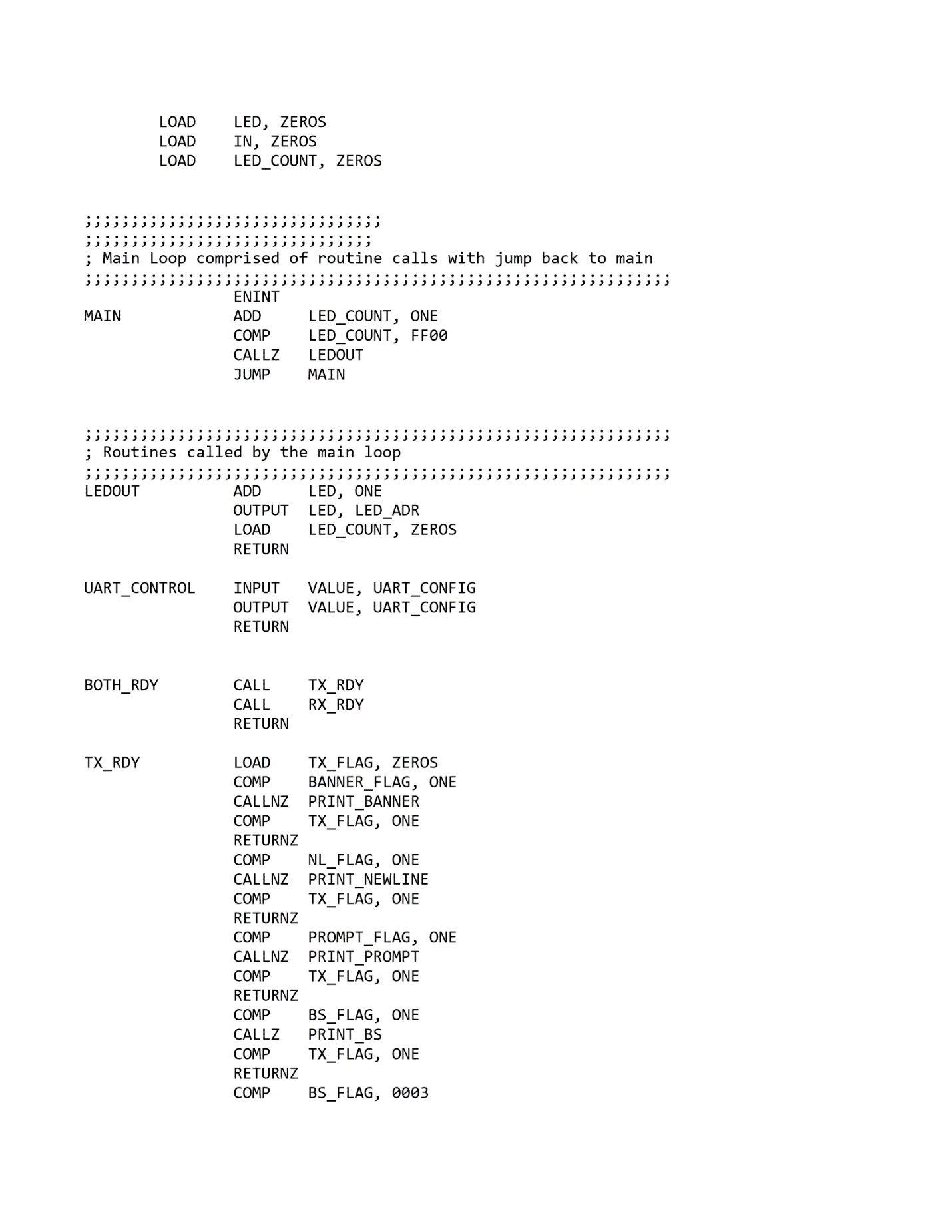


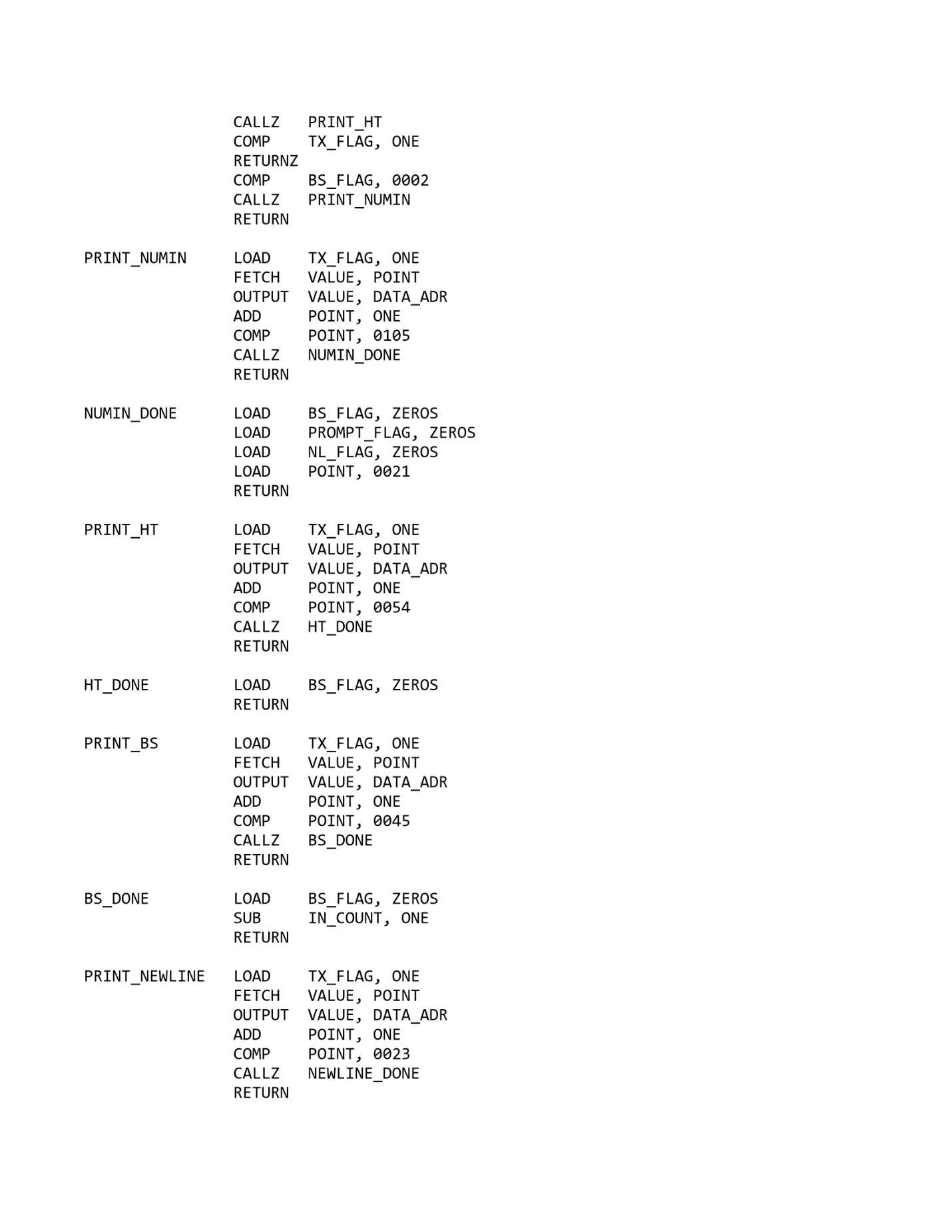


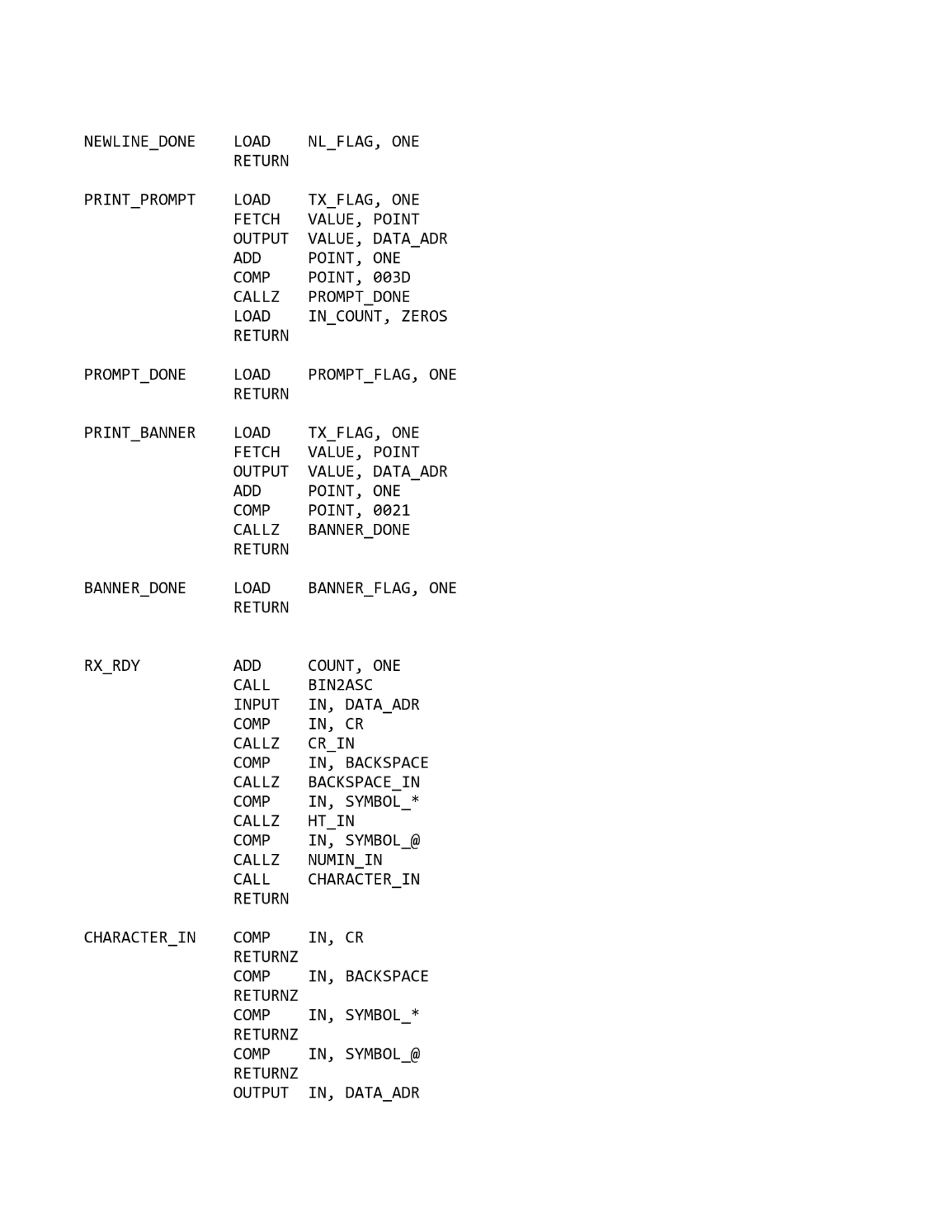


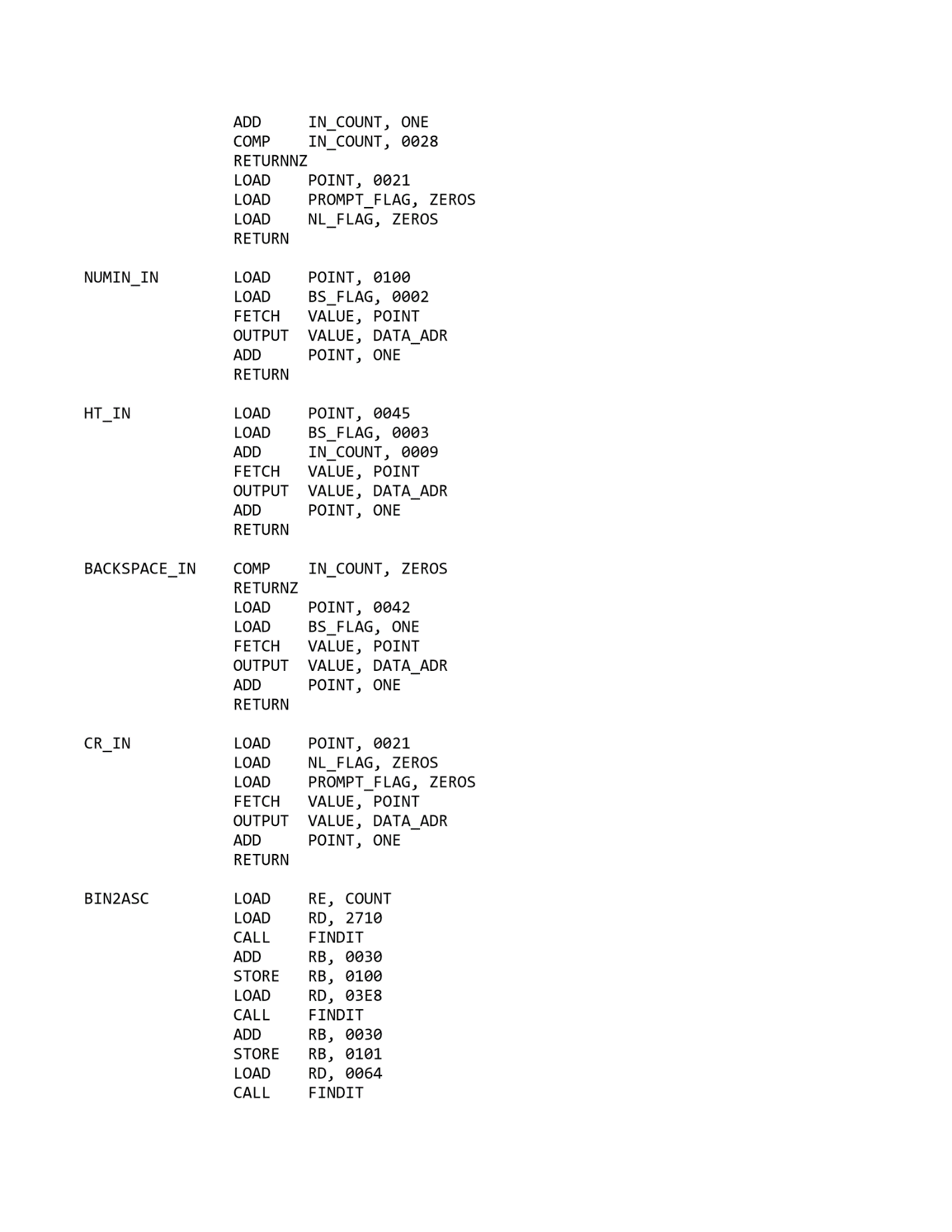


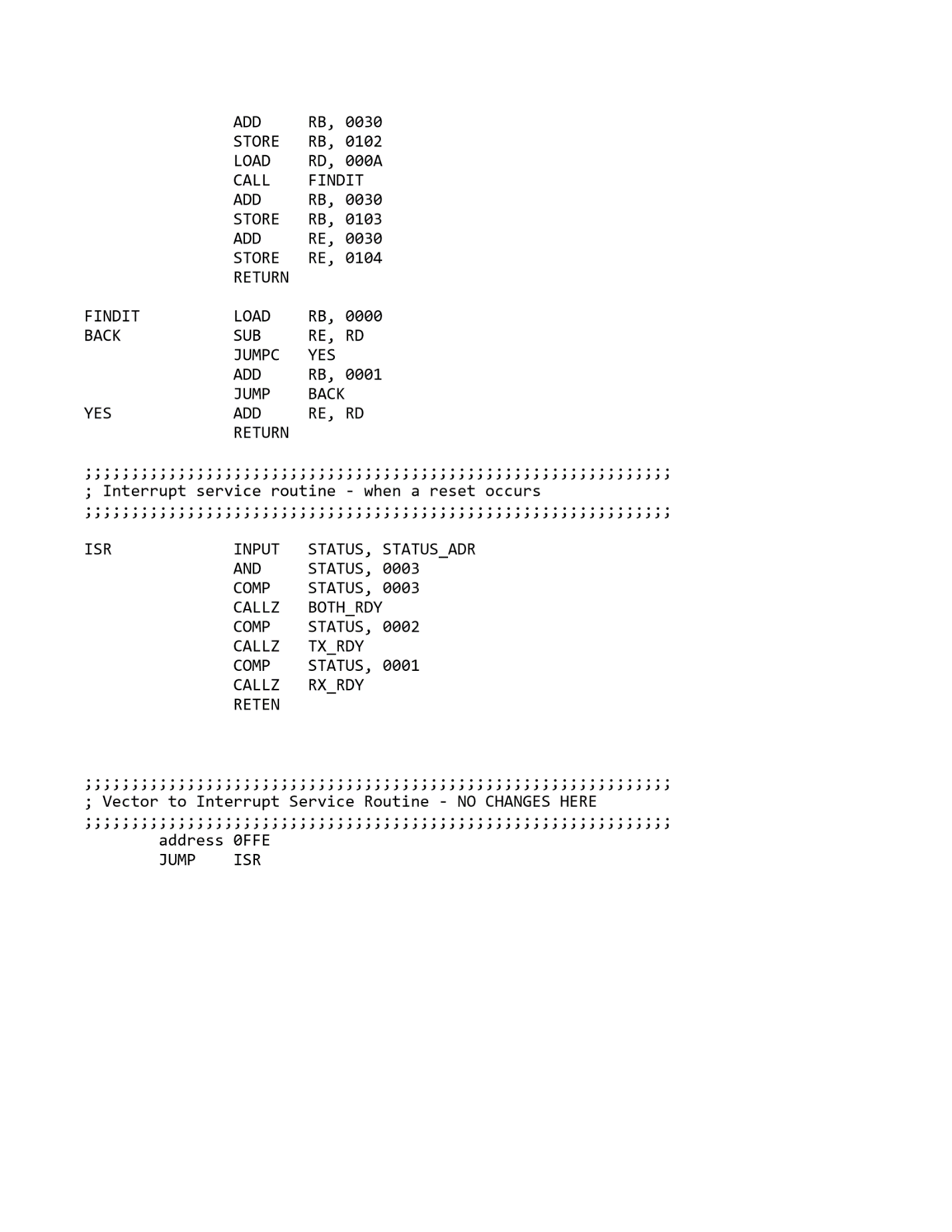




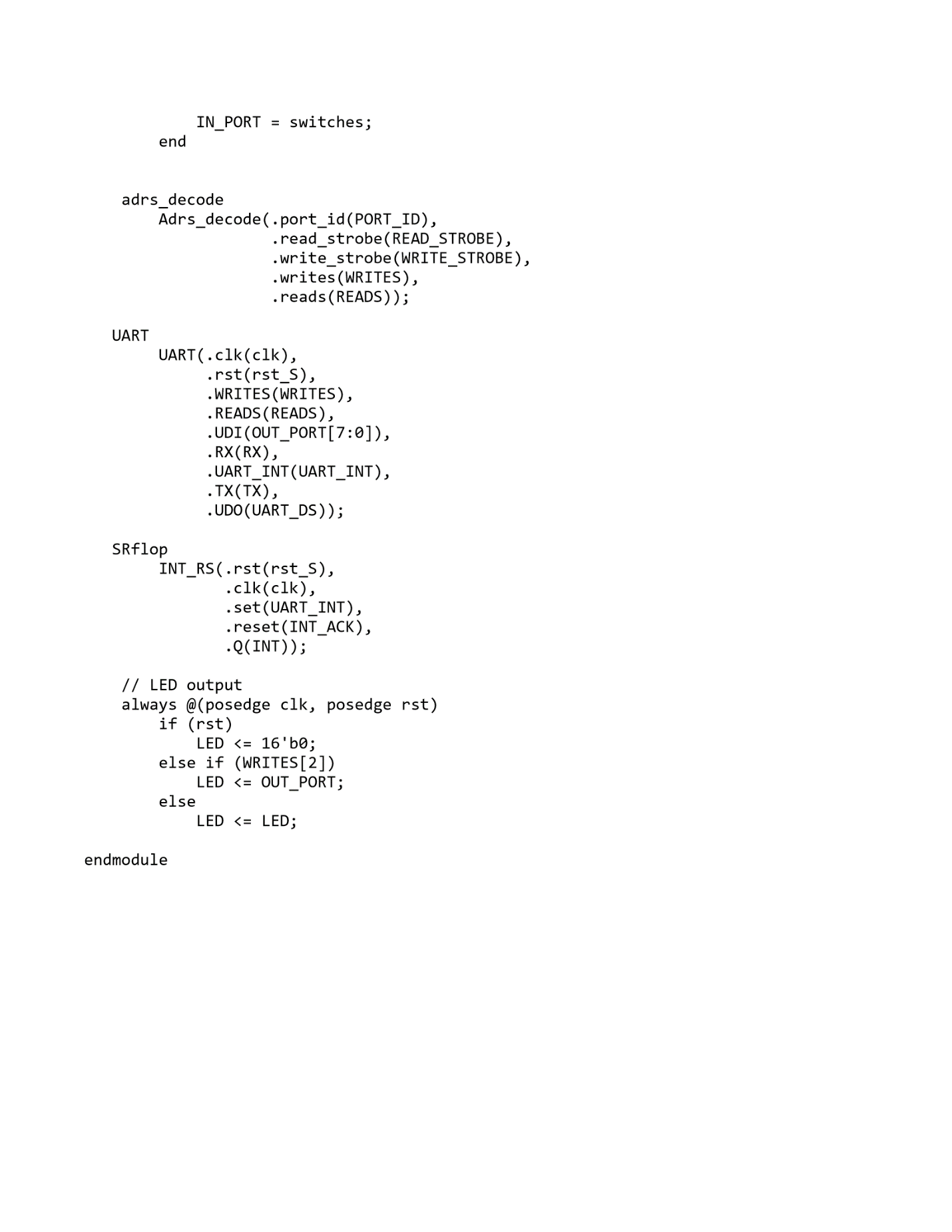




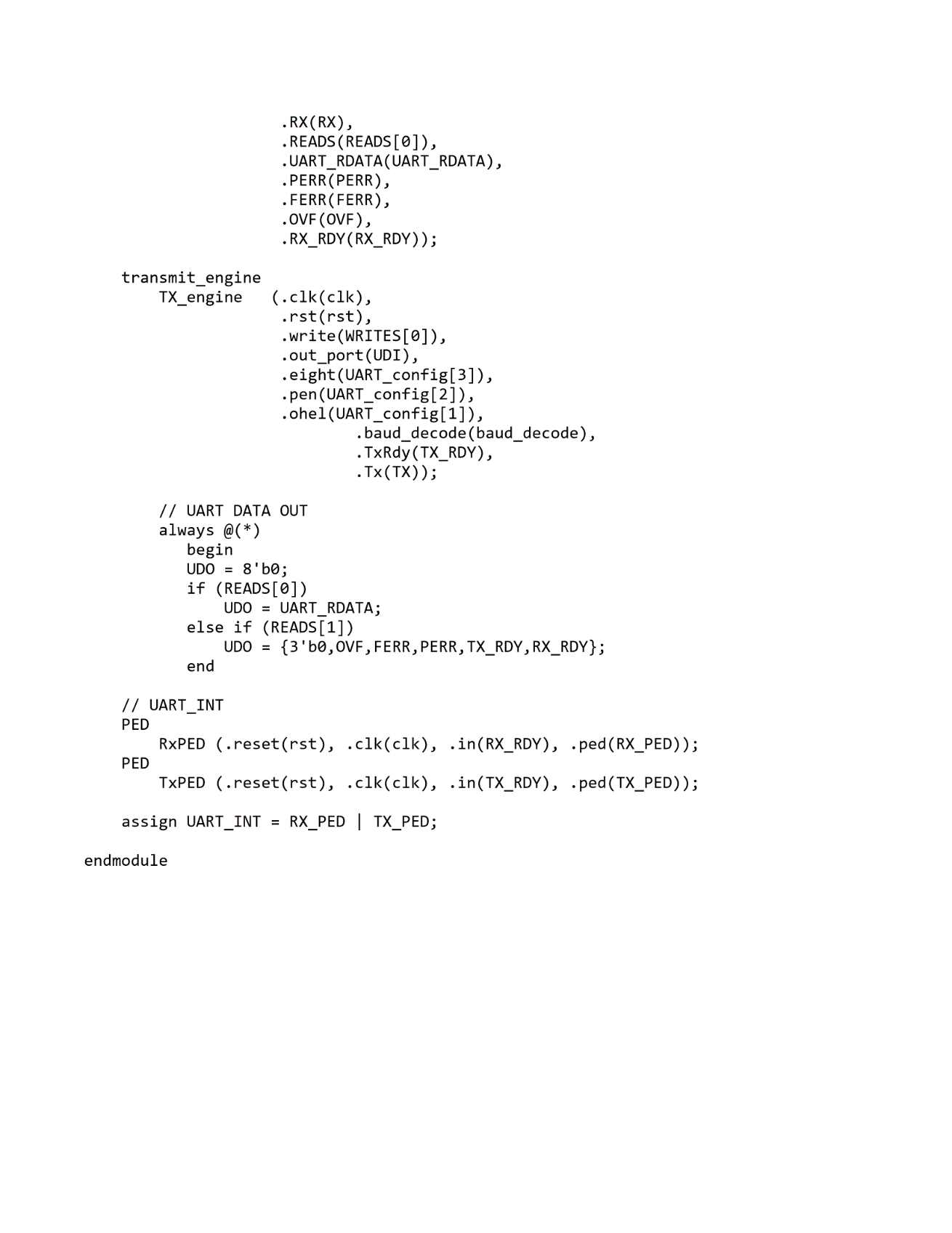




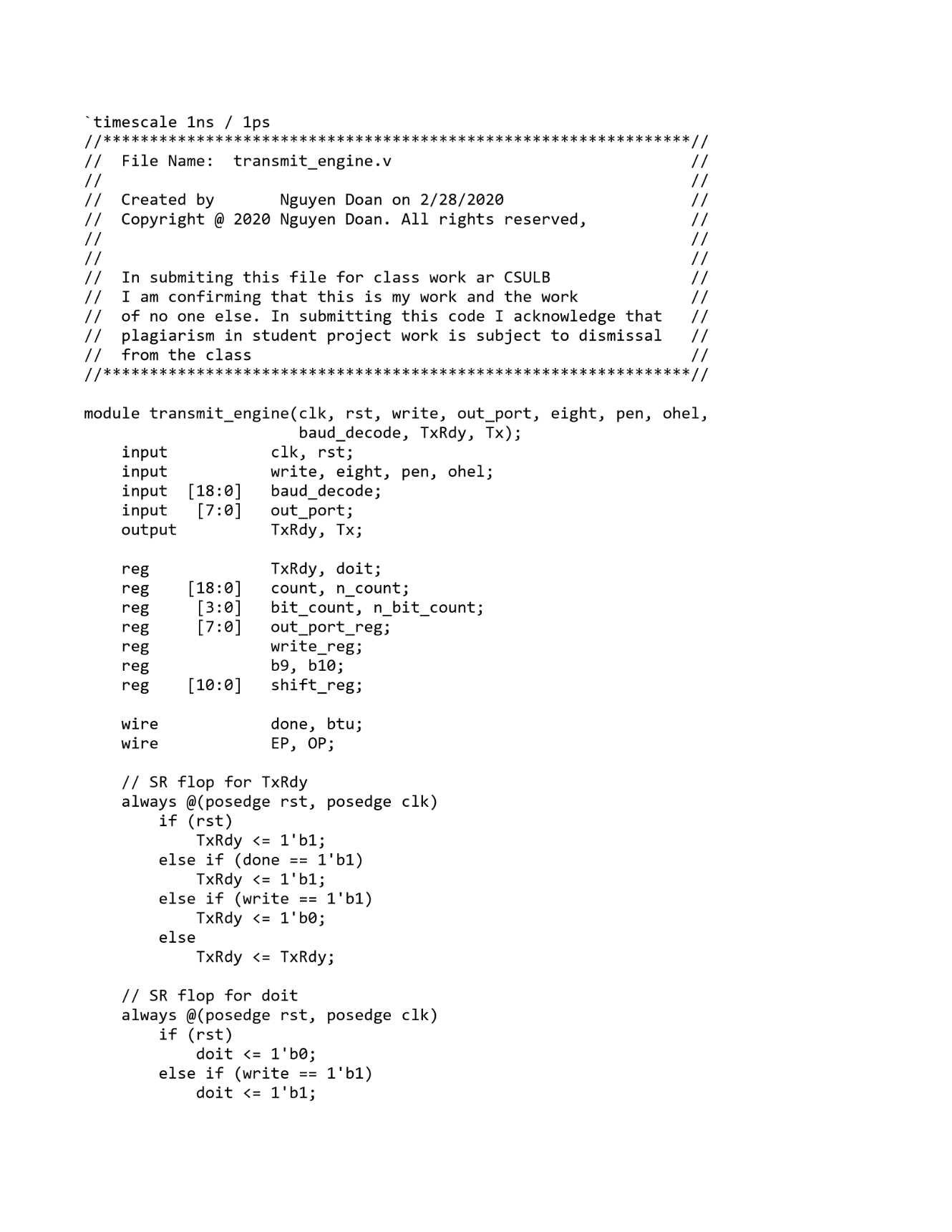
## Appendix 2: Top Level Design

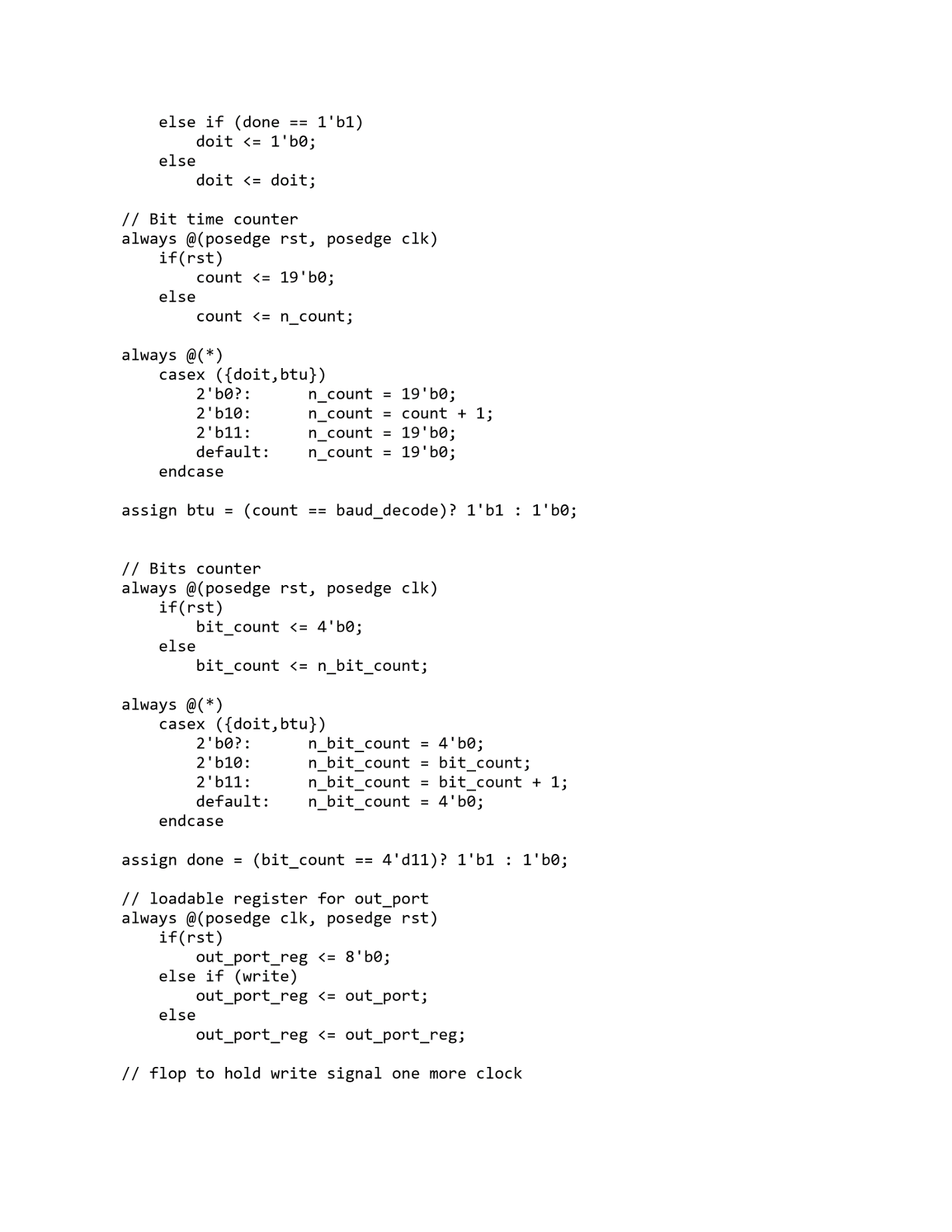


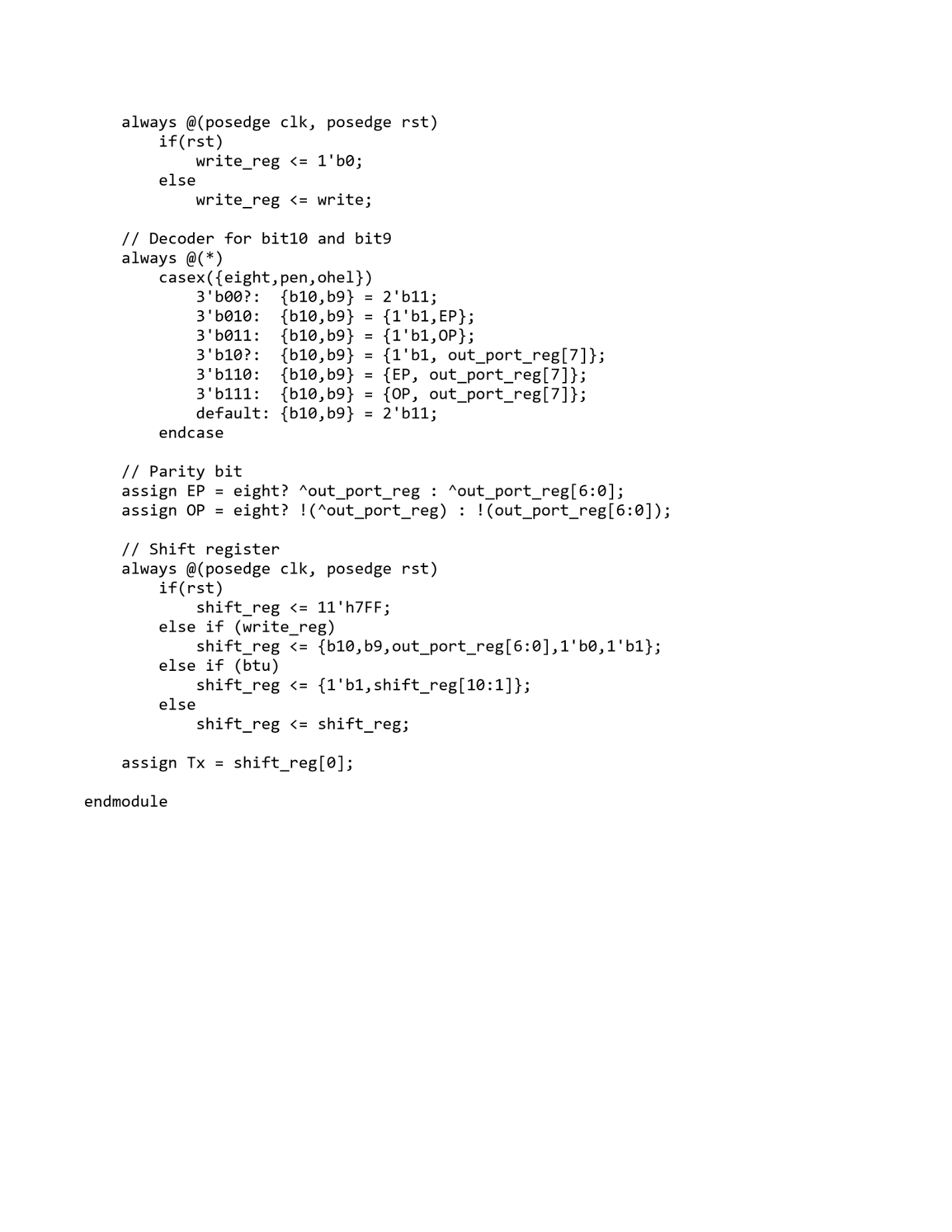
## Appendix 3: UART



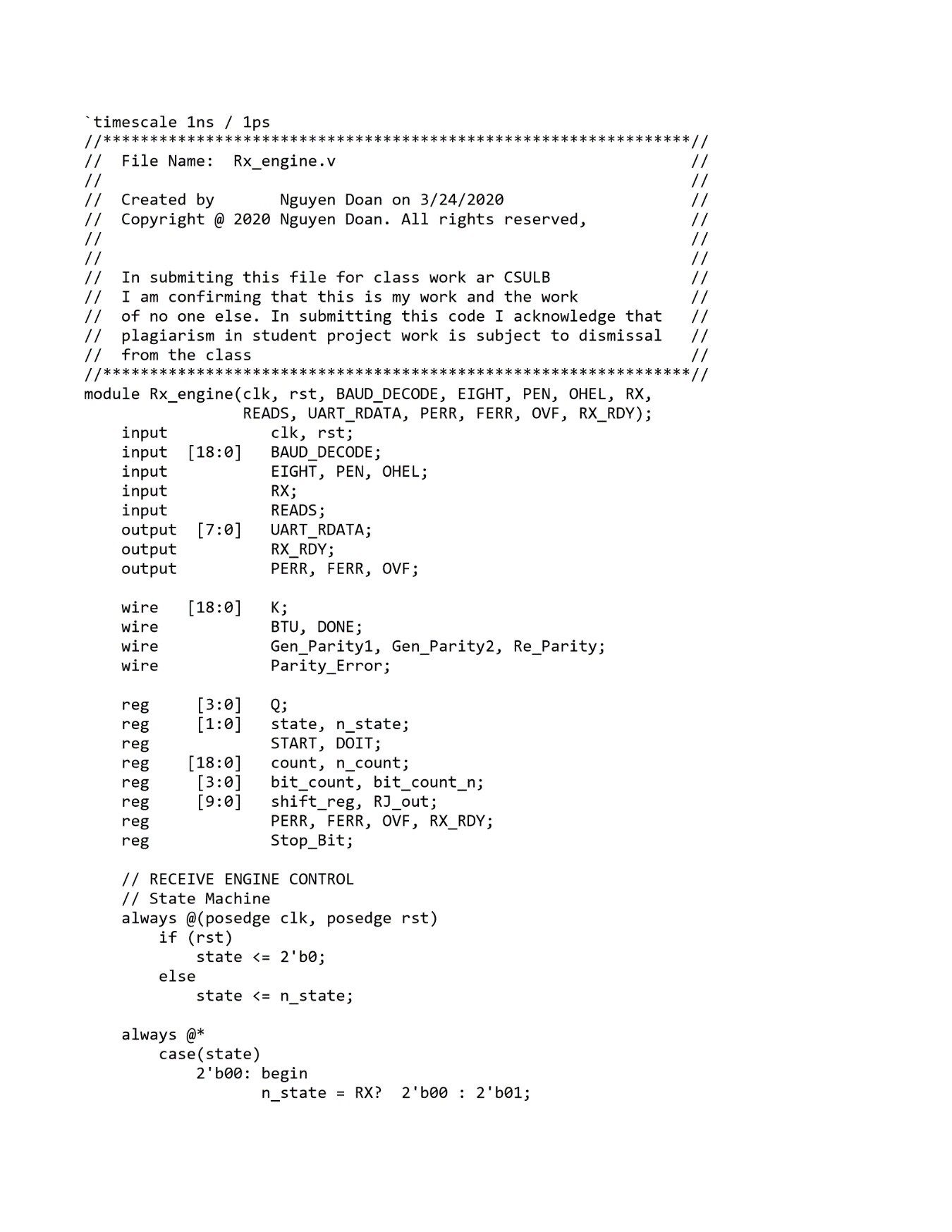
## Appendix 4: TX\_Engine

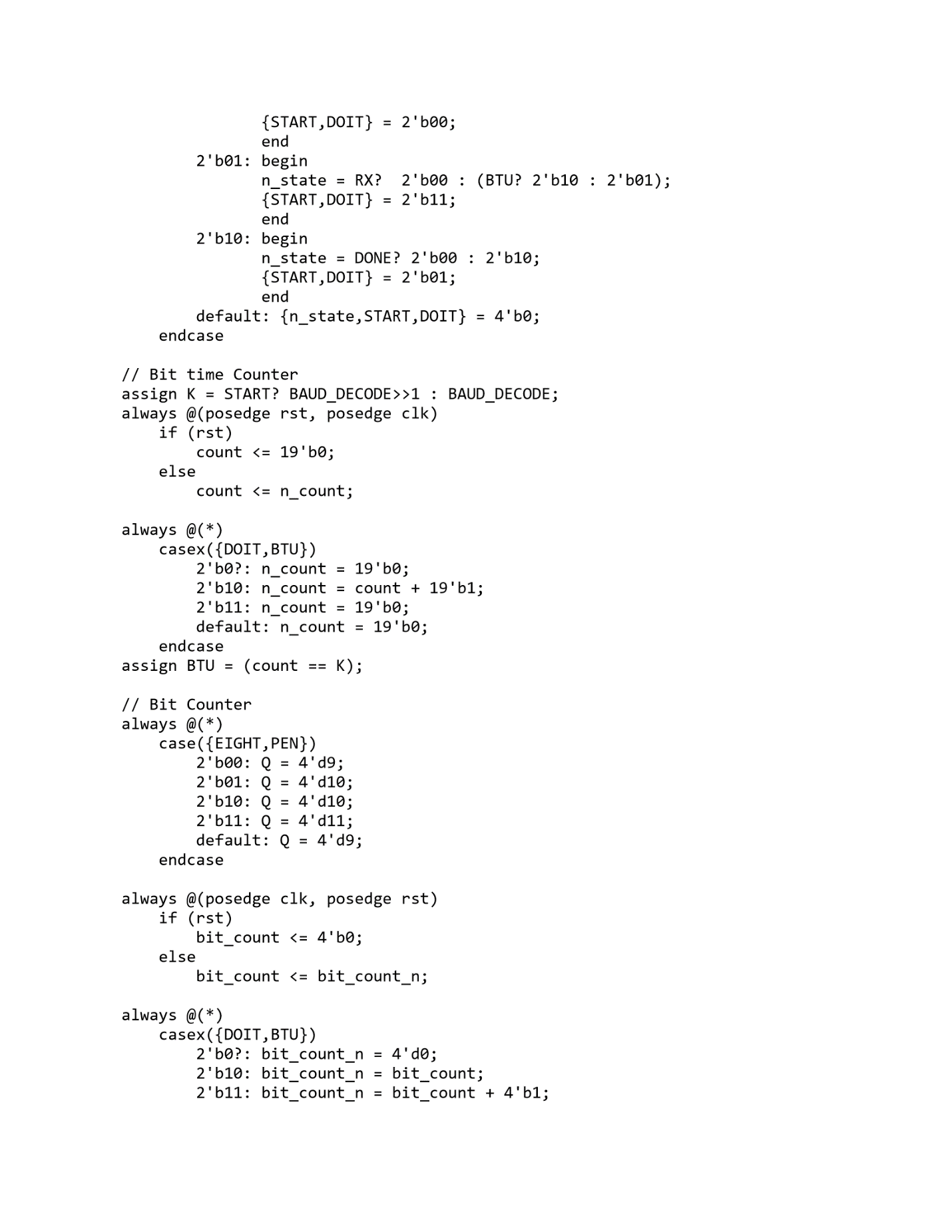


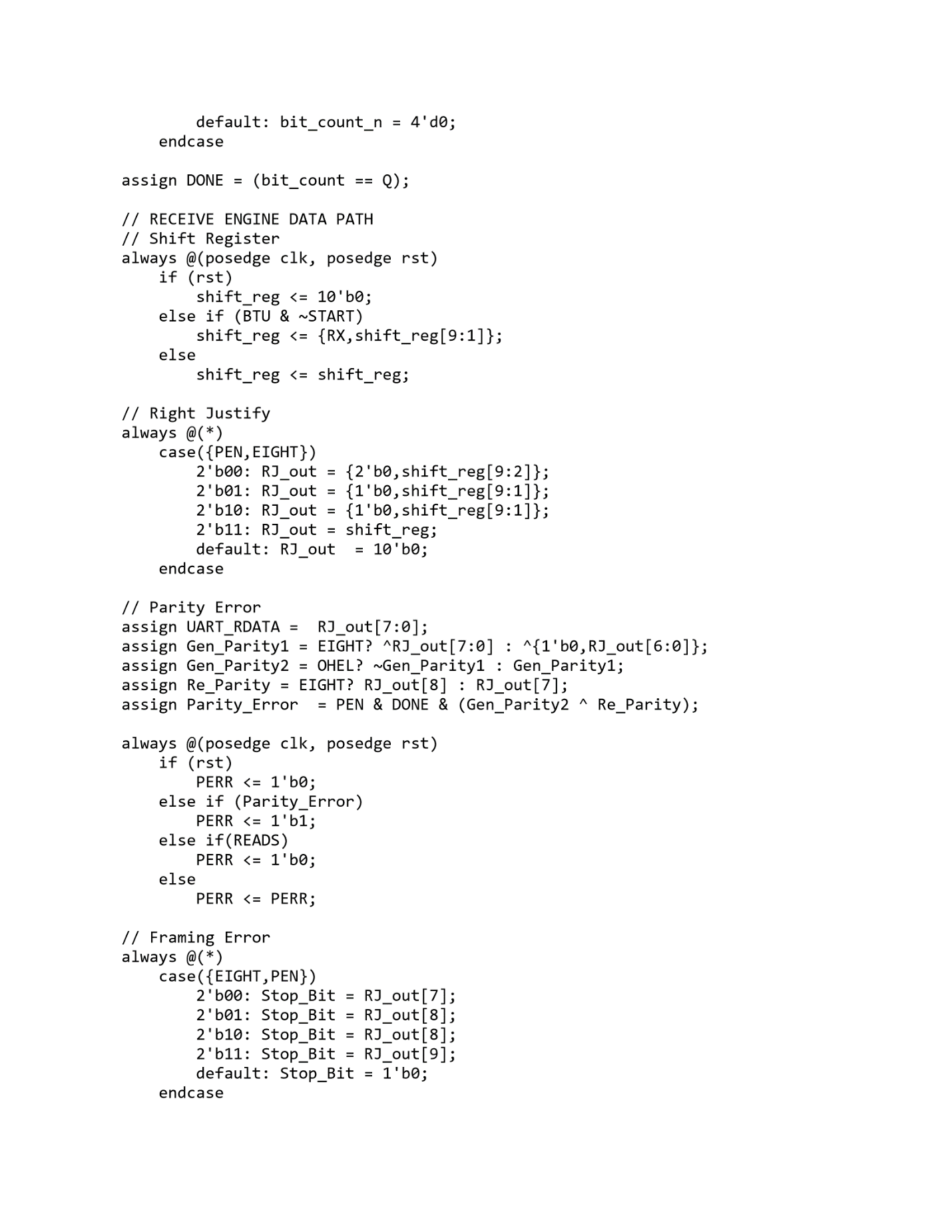


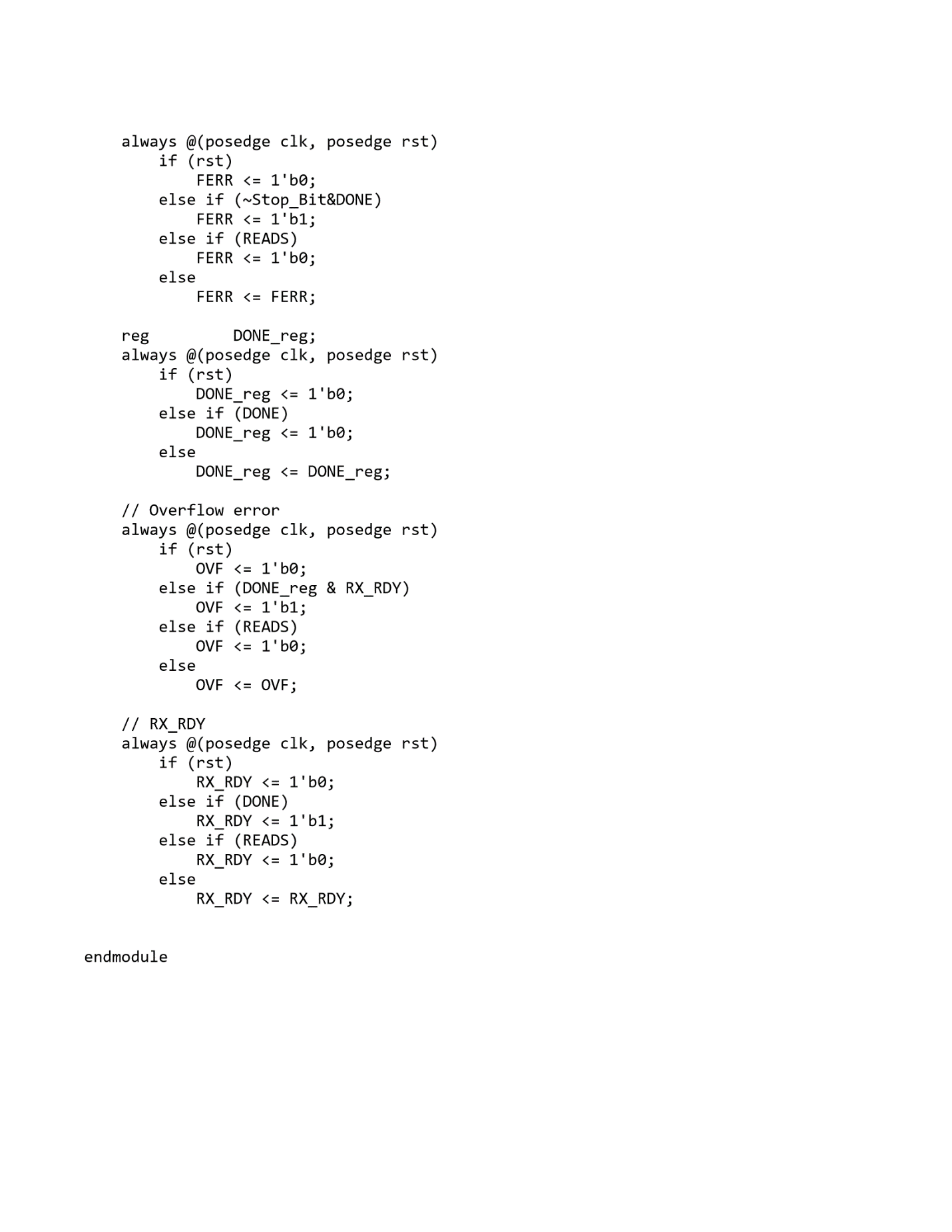


## Appendix 5: RX\_Engine

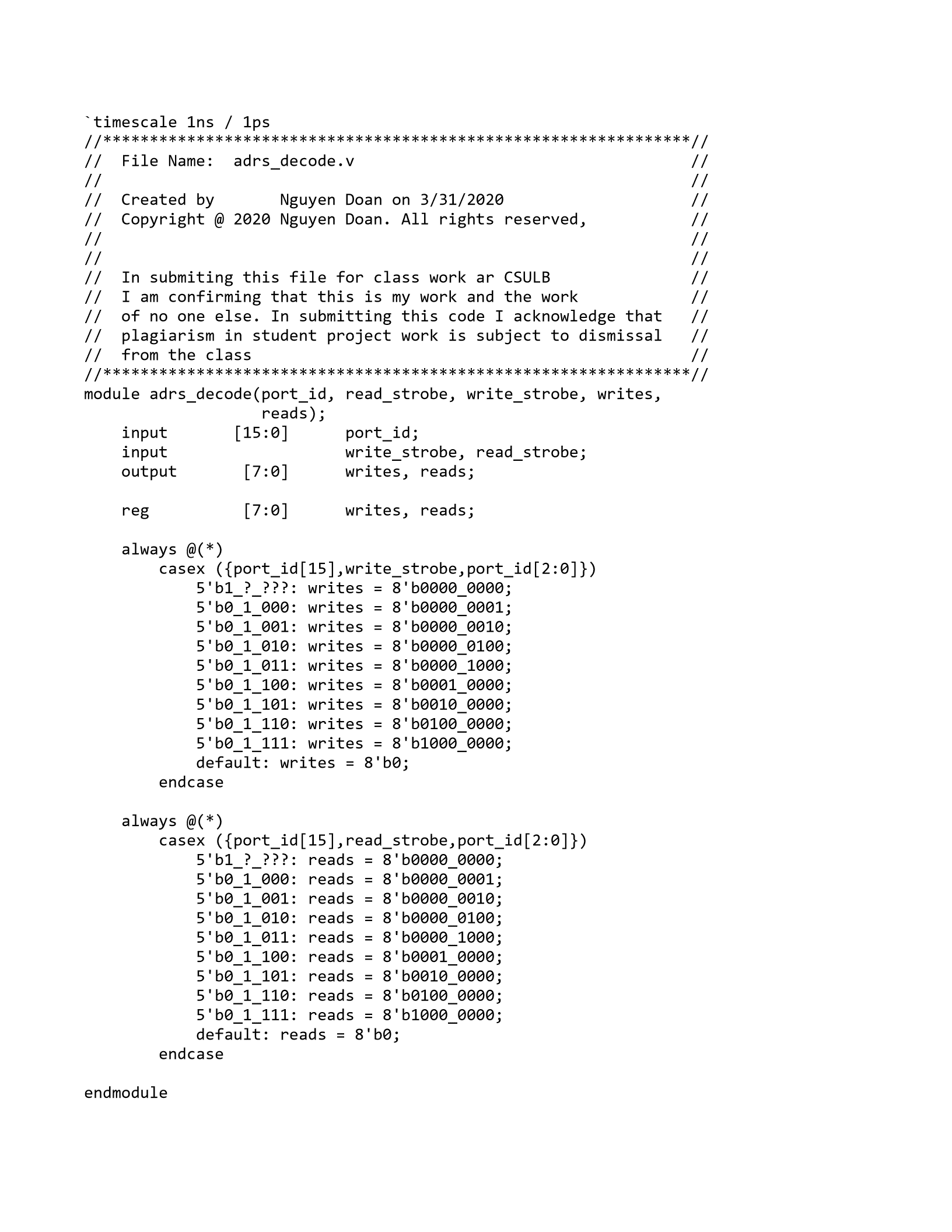








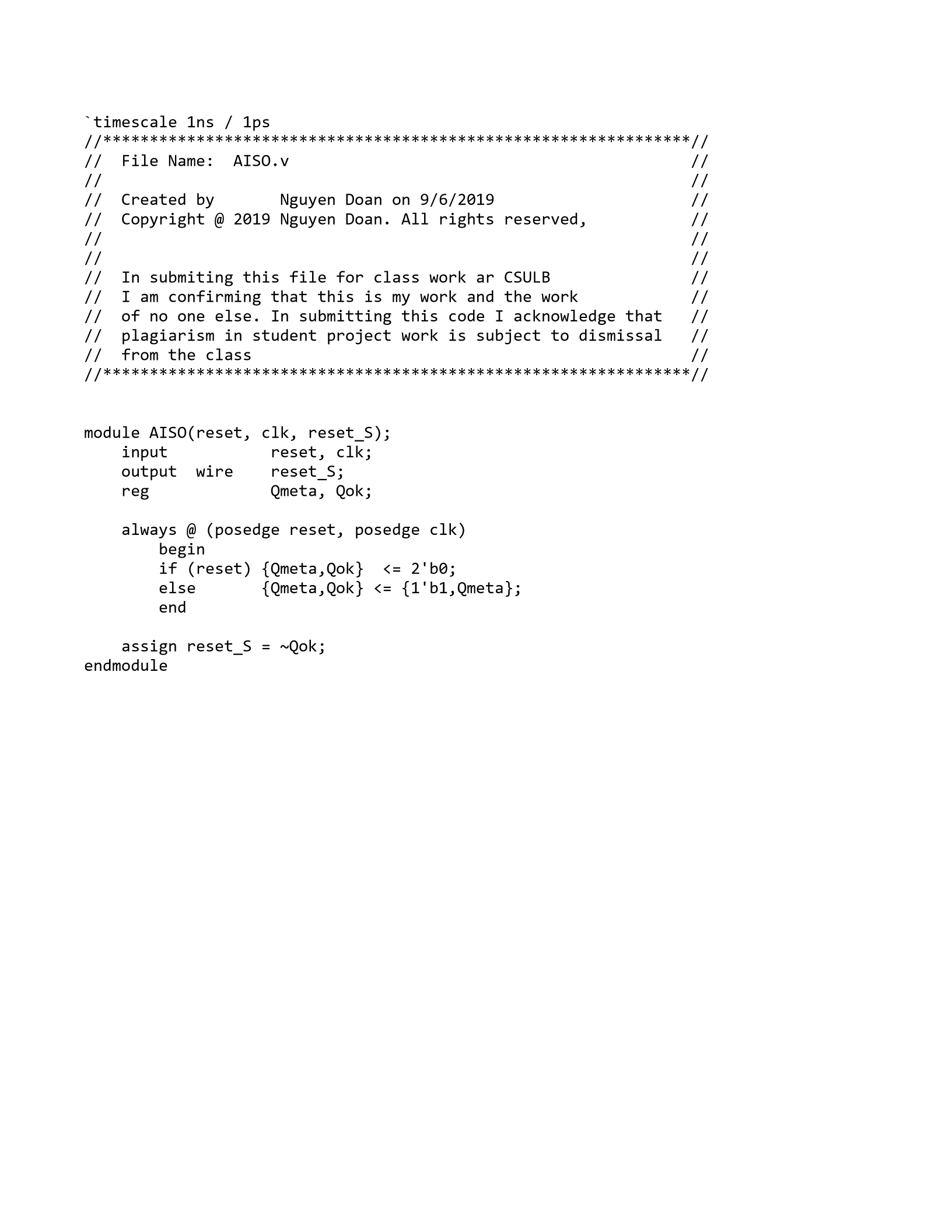
## Appendix 6: Address Decode



## Appendix 7: Baud Decode

## Appendix 8: PED

## Appendix 9: AISO



## Appendix 10: SRflop

