Raspberry Pi Object code calculator
Elian Fernandez Borboa
823256194

This is the README file for your very own raspberry pi object code calculator . For this project it is necessary to have a raspberry pi, a breadboard , an I2C 16 x 2 display, a keypad membrane, a button , a resistor and some jumper cables to connect pins together.

For this project you will need:.
RPi_I2C_driver library for LCD display
RPi.GPIO library for GPIO handling

Step1 : you want to install
The required GPIO libraries for the handling of the GPIO pins on the raspberry pi

Step 2: assemble the prototype , first and foremost you must assemble all of the components i had listed and also refer to the raspberry pi diagram provided in the pdf.

All raspberry pis have 40 pins please refer to your user manual on what each pin is labeled as , you will need to reserve 8 GPIO pin slots for the keypad membrane , for the i2c display you will need another 4 pins these pins are a ground pin , a vcc pin, an sdl pin, and an sda pin. For the button you will only need one 3v3 pin going from one side of the button on the breadboard to the pi and another pin going from another side to any GPIO pin

In assembling the pins you need to take note on the 8 GPIO pins for the keypad membrane also it is important to note that going from left to right the first 4 cables on the keypad will be the rows and the next 4 will be the columns this is crucial for the keypad to work so take note individually on which cable is going to which GPIO pin and don't forget to add a resistor to the button.

This is a handy code snippet that will help you keep track of those GPIO pins
Fill this out from top to bottom L1 being the input furthest to the left and C4 being the rightmost input finally the button pin will be your GPIO for the button
L1 =
L2 =
L3 =
L4 =
C1 =
C2 =
C3 =
C4 =
button_pin =

Step 3: once you have successfully assembled the pi and its accessories we must now open up our terminal we must make sure that our RPi_I2C_driver.py file is in the same directory as our project.py file. We will run the following command

Python3 RPi_I2C_driver.py

This command will make sure that we are able to use the library of functions provided within this file

Next we will have to open up our project.py file. Our first changes will be to change the pins as mentioned above this starts at line # Define pins once this is done there is nothing else to do other than to try out our new handy dandy Object code calculator

Step 4 : I forgot there are some customizable features based on user preference this will also explain some functions and their

The following lines can be changed to display a different prompt

```
def display_prompt(): # this function will display my prompt
   mylcd.lcd_display_string("Enter OBJcode:", 1)
```

The following lines can also be changed to map the keypad to your desired values for example i swapped out “#” and “*” for E and F

```
display_entered_code(received_code)
        received_code = read_line(L1, ["1", "2", "3", "A"], received_code)
        received_code = read_line(L2, ["4", "5", "6", "B"], received_code)
        received_code = read_line(L3, ["7", "8", "9", "C"], received_code)
        received_code = read_line(L4, ["F", "0", "E", "D"], received_code)
```

The following lines can be changed to change the delay of how long the lcd displays the output for

```
time.sleep(8)  # displays output for 8 seconds before clearing screen
            mylcd.lcd_clear()
```

Functions:

display_prompt(): Displays the initial prompt on the LCD.

display_entered_code(code): Shows the entered code on the LCD.

read_line(line, characters, received_str): Reads the input from the membrane keypad.

Other functions include the logic handling of the object code:

get_xbpe(temp_opcode) Determines the addressing mode and whether it is indexed or not based on the input opcode.

determine_format_3_or_4(temp_opcode) Determines whether the input opcode corresponds to format 3 or format 4.

get_nibits(hex_string) Extracts and returns the n and i (Immediate, Indirect, or Simple) from the given hexadecimal string.

get_mnemonic(hex_string) Retrieves the mnemonic instruction based on the input hex string.

check_and_get_mnemonic(input_string) Checks if the input string corresponds to a format 2 instruction and retrieves its mnemonic if it exists.