# Project1 Report

118010317

# Instruction

My project is composed of three source files and one testing file.

phase1.py

labelTabel.py

phase2.py

test.py

# Phase 1.py & labelTable.py

This part mainly to read the input file, delete useless comment or symbols, leaving only necessary code for machine to read. The output of this part will be two dictionary: Instr\_dict{} and label\_dict{}. The keys in label\_dict are labels shown in input file, the values are the corresponding address of the label. The keys in instr\_dict are address of instructions, and the values are strings of instructions. Remark: the first address is binary of 0x400000 as mentioned in the assignment requirement.

Eg. instr\_dict = {address , ‘add $t0, $t1, $t’}

Eg. Label\_dict = {Fibonacci, address}

Implementation

1. delete\_comment() : delete the comments ‘’’ ‘’’ and ###
2. get\_label() : return the label, else will return ‘’
3. get\_instr() : return the instruction
4. hex2bin() : hexadecimal to binary
5. set\_instr() : read input file, return Instr\_dict{} and label\_dict{}

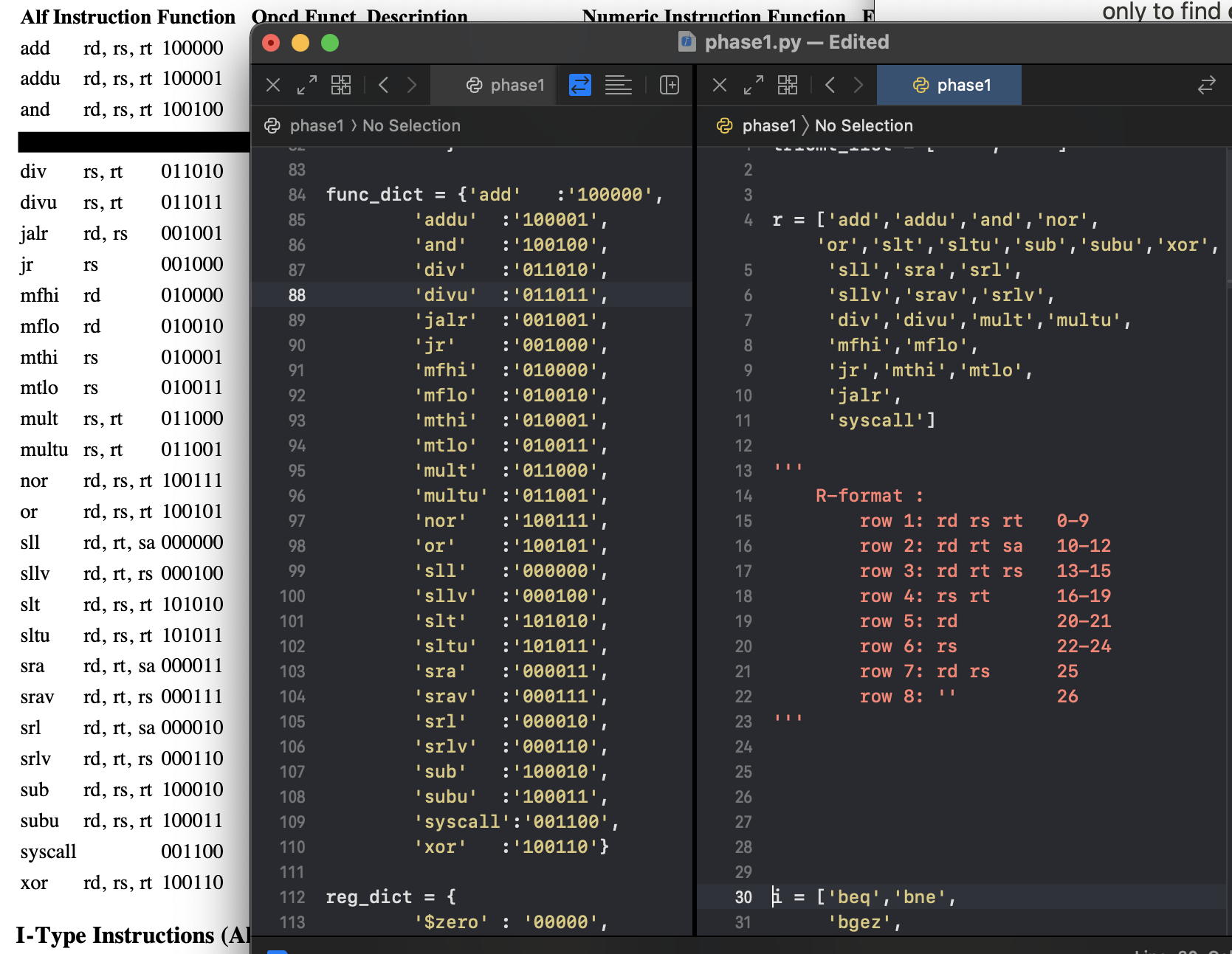
# Phase2.py:

### R-Type instructions:

According the book, the structure of machine code for an R-format instruction is fixed, the output machine code is expected to have the form:



Therefore, in order to output the machine code for an R-format instruction, we need only to find **opcode, rs, rt, rd, sa, function,** respectively. All R-type instructions use a 000000 **opcode**. The operation is specified by the function field. code for **function** is stored in a dictionary, as shown below.The tricky part is to find **rs, rt, rd, sa.** To do so, I have classify the R-type format instruction into 8 categories. As shown below, for example, the original instruction form in first line in r\_list should be opc rd rs rt. After defining the type in r-format, I will be able to get rs, rd, sa from the original instruction.

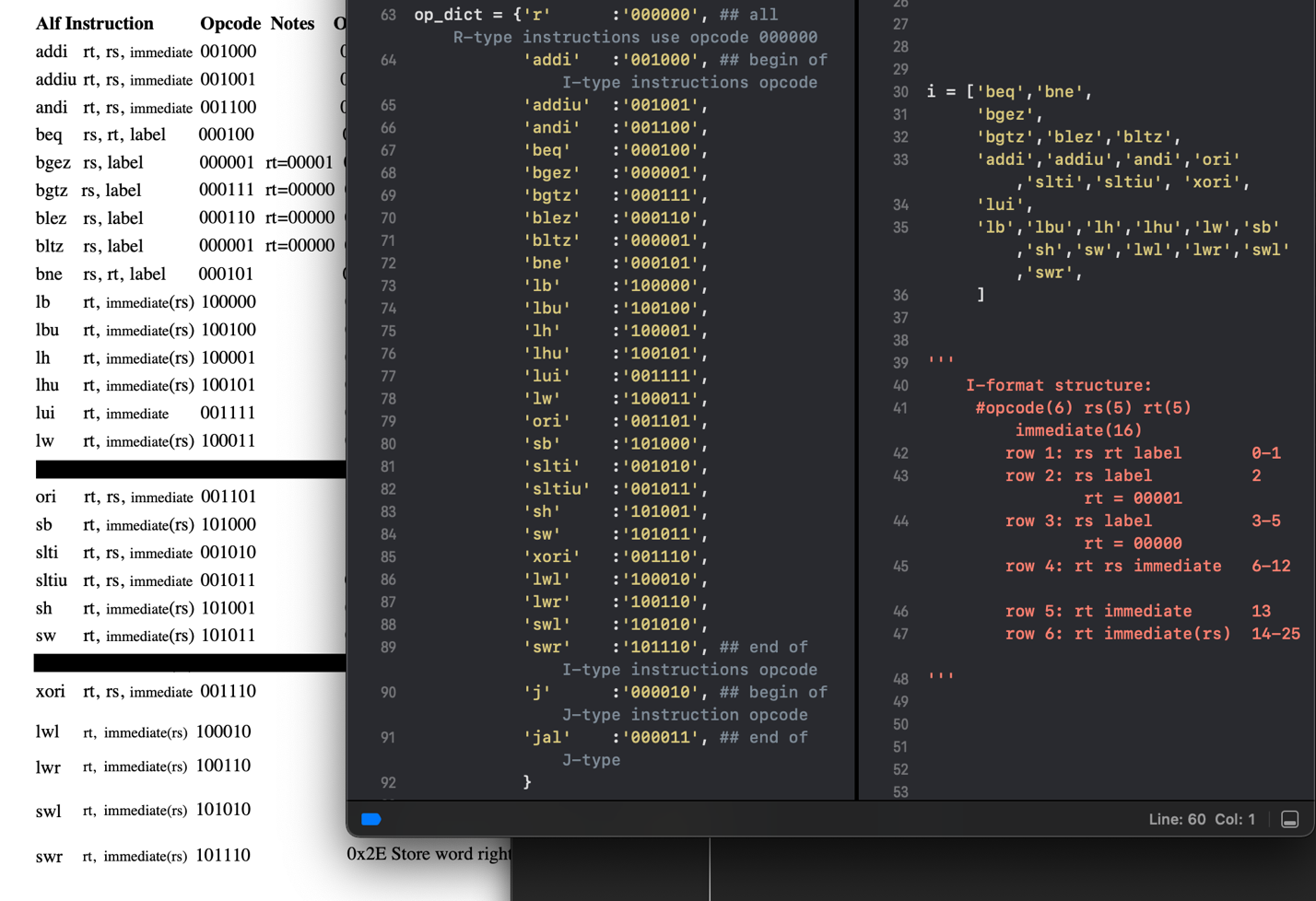


### I-type Instruction:

According the book, the structure of machine code for an I-format instruction is fixed, the output machine code is expected to have the form:

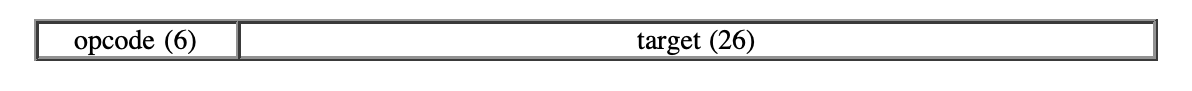


Therefore, in order to output the machine code for an I-format instruction, we need only to find **opcode, rs, rt, rd, immediate** respectively. Code for **function** is stored in a dictionary, as shown below. The tricky part is to find rs, rt, immediate**.** To do so, I have classify the I-type format instruction into 6 categories. As shown below, for example, the original instruction form in first line in r\_list should be opc rs rt label. After defining the type in i-format, I will be able to get **rs, rt, immediate, label** from the original instruction. However, note that label should be translated to immediate value(16 bit).

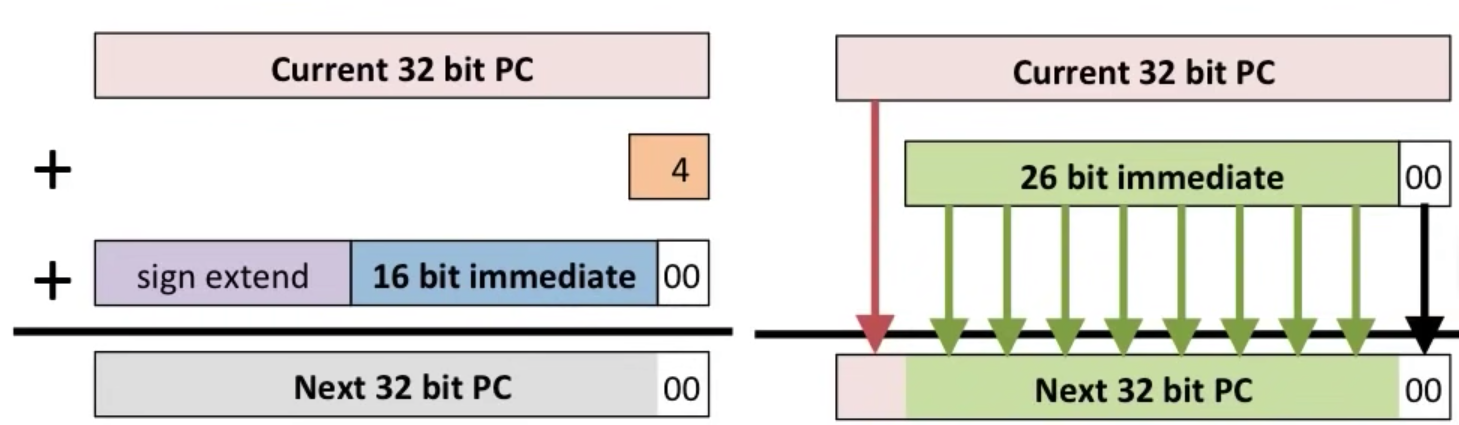


### J-type Instruction:

According the book, the structure of machine code for an J-format instruction is fixed, the output machine code is expected to have the form:



Therefore, in order to output the machine code for an I-format instruction, we need only to find **opcode, target** respectively. Code for **function** is stored in a dictionary, as shown below. The tricky part is to find target (26 bit)**.**



Implementation:

1. classify\_type() : return the instruction type: R or I or J
2. get\_rstrc() : return the classification result in R-type
3. get\_istrc() : return the classification result in I-type
4. get\_op() : return the opcode of R-type, I-type, J-type respectively
5. get\_rs() : return register rs in R-type, I-type repectively
6. get\_rt() : return register rt in R-type, I-type repectively
7. get\_rd() : return register rd in R-type
8. get\_shamt() : return shift amount of R-type
9. get\_func() : return function in R-type
10. get\_imdt() : return immediate value(16 bit) in I-type
11. get\_target() : return target value(26 bit) in J-type
12. num2bin() : return positive integer number to binary case
13. neg2bin() : return negative integer number to binary case

# Testing:

I use both the .asm file and get the output .txt file. After comparing with the expected .txt file, I have passed the test and get all the answers right.