浙江大学

题　　目	汇编器实验报告
课程名称	计算机组成
授课老师	刘海风
学生姓名	肖瑞轩
学　　号	3180103127
专　　业	混合班
学　　院	竺可桢学院

计算机组成实验——汇编器与反汇编器设计

3180103127　混合班　肖瑞轩

一、汇编器实现的基本功能

基本的功能与extenton

（1）实现了基本的汇编与反汇编，包括了18条基本的指令：Add, Sub, And, Or, Addi, Ori, Sll， Srl，Lw, Sw, Lui, Slt, Slti, Beq, Bne, J, Jal, Jr.得到实现

(2）Extension1:实现了6条MIPS伪指令，Bgt, Bge, Blt, Ble, move,not,clear

(3) Extension2:实现了一些其他更加复杂的MIPS指令，如：Xor, Nor, Sra, Xori,SW, Jalr, sllv, srlv, srav，lhx, lh, lwx，lhux, lhu, shx, sh, swx,mul,mult,div,mfhi,mflo,mthi,mtlo以及系统指令eret,syscal等丰富而众多的指令。

(4）Extension3:使用tkinter库进行了基本的可视化与ui交互界面的设计，方便用户进行操作。

其他的一些功能：

（1）支持直接打开文件或者采用直接在文本窗口进行打字输入，也可以直接在文件的基础上进行编辑修改。

（2）支持即时在文本框中进行模仿ide的修改，ctrl+c/v的赋值粘贴操作与即时的汇编、反汇编操作。

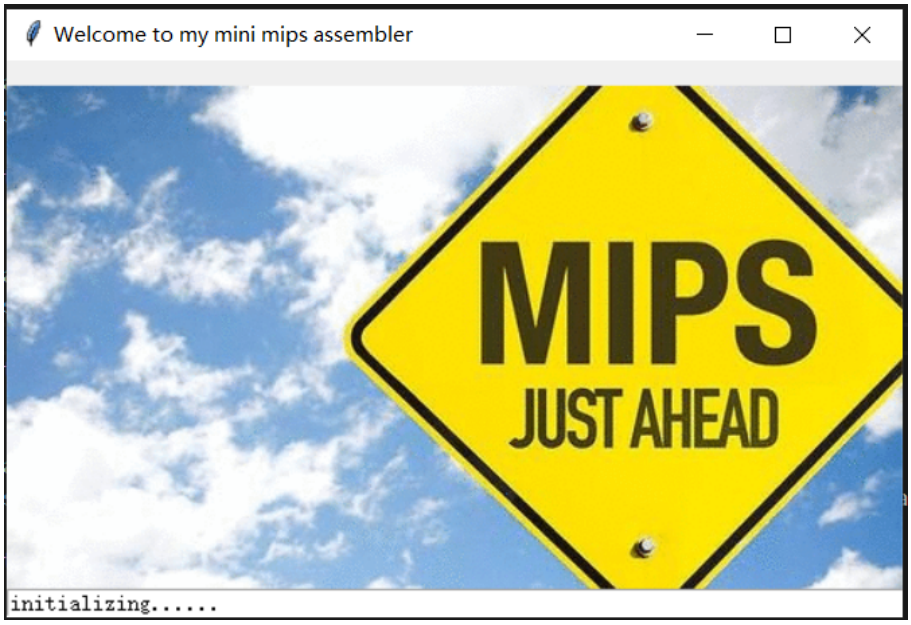
（3）支持机器码二进制与十六进制的即时转换与显示，支持打开文件中注释的中文读取。

（4）支持保存为或打开格式为.txt,.bin,.coe等等符合格式要求的文件。

（5）加入了应用程序图标、入场动画、主页面图形、帮助提示等等

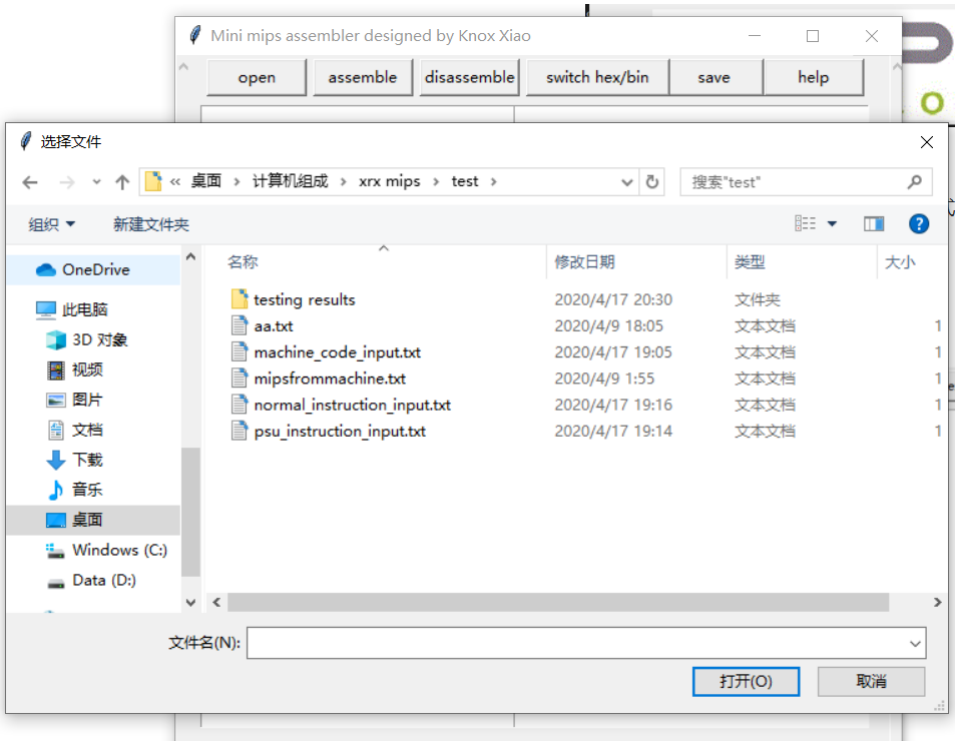
二、汇编器的操作说明与演示

在打开时设置了开启的动画与欢迎语

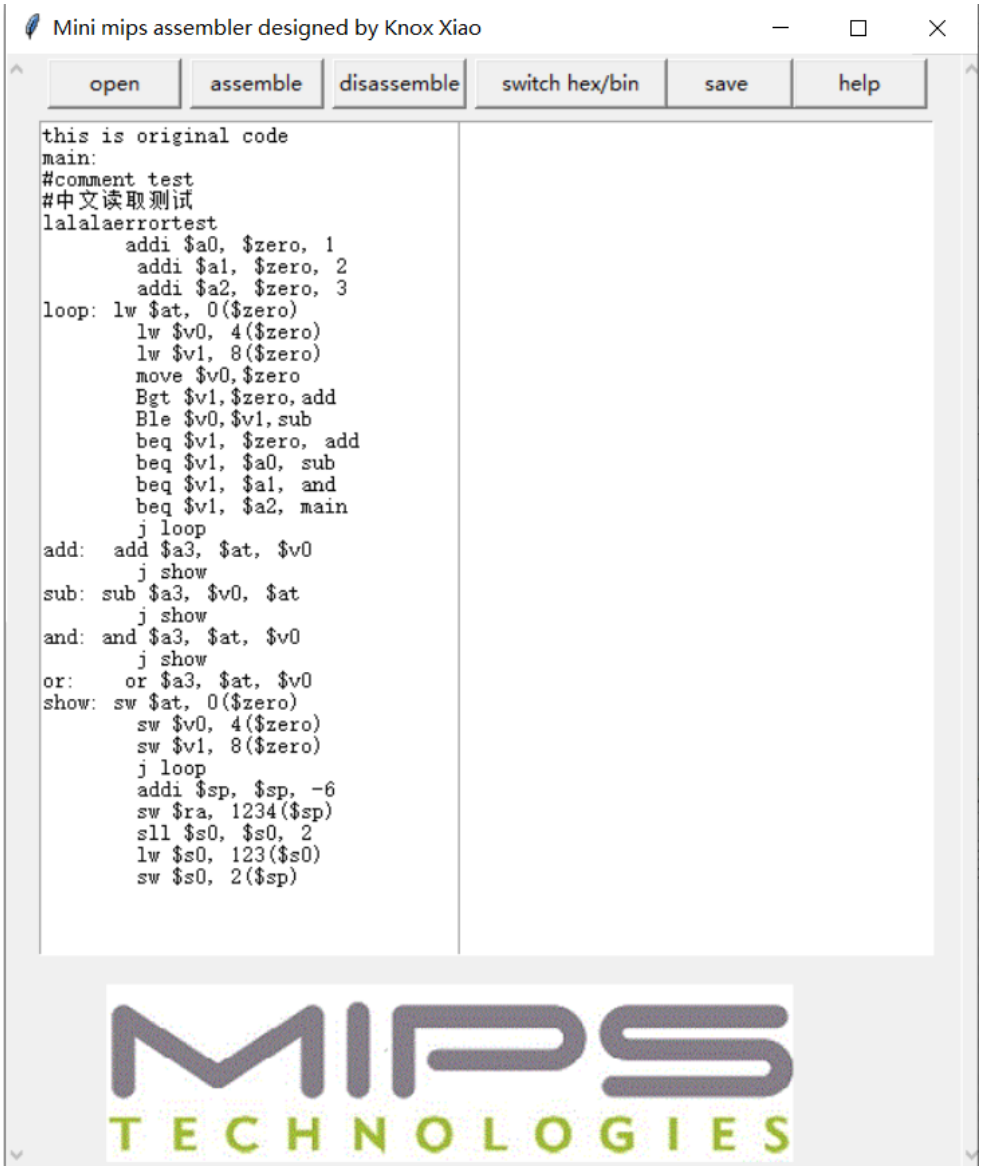


进行汇编操作的主页面：

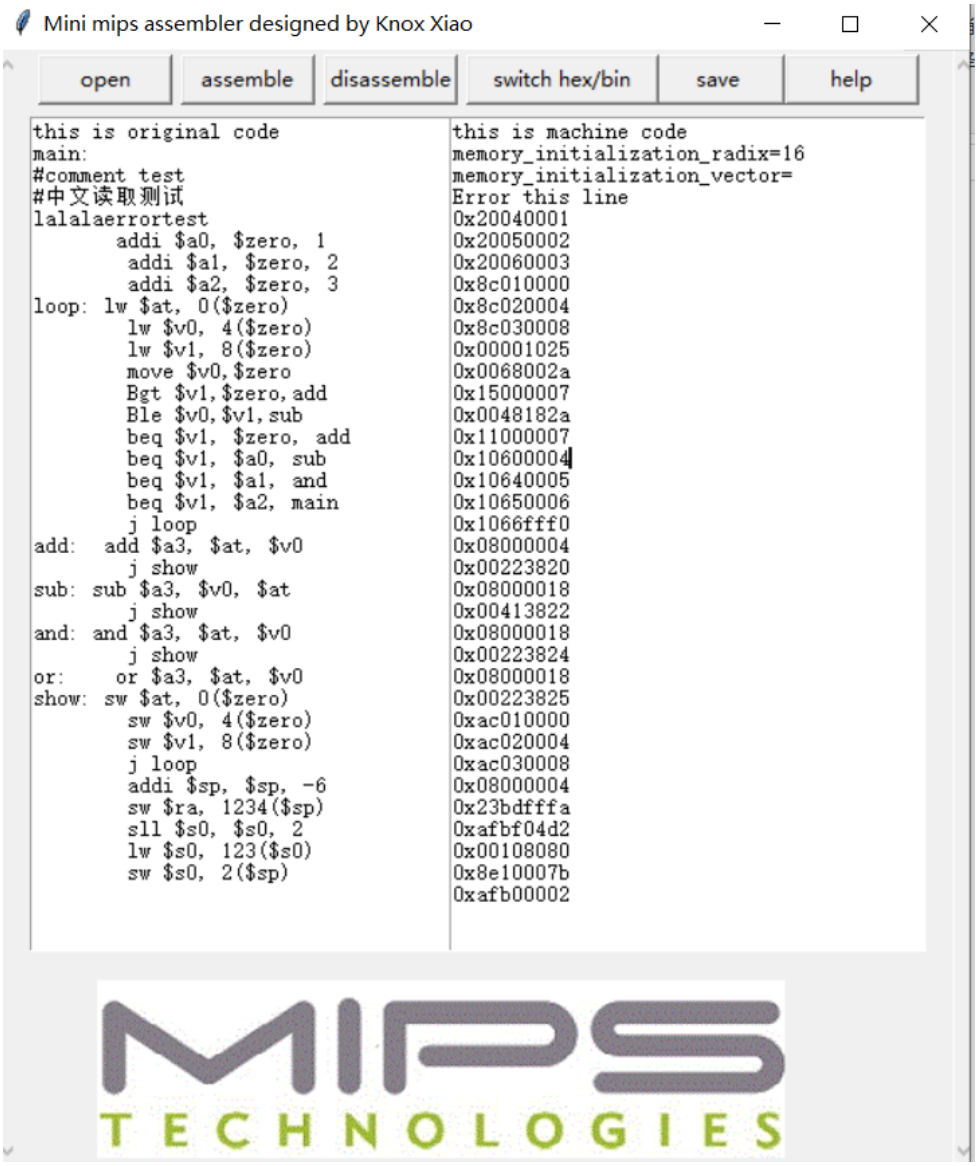
点击open按钮可以选择文件进行打开（支持.coe .bin .txt等格式），



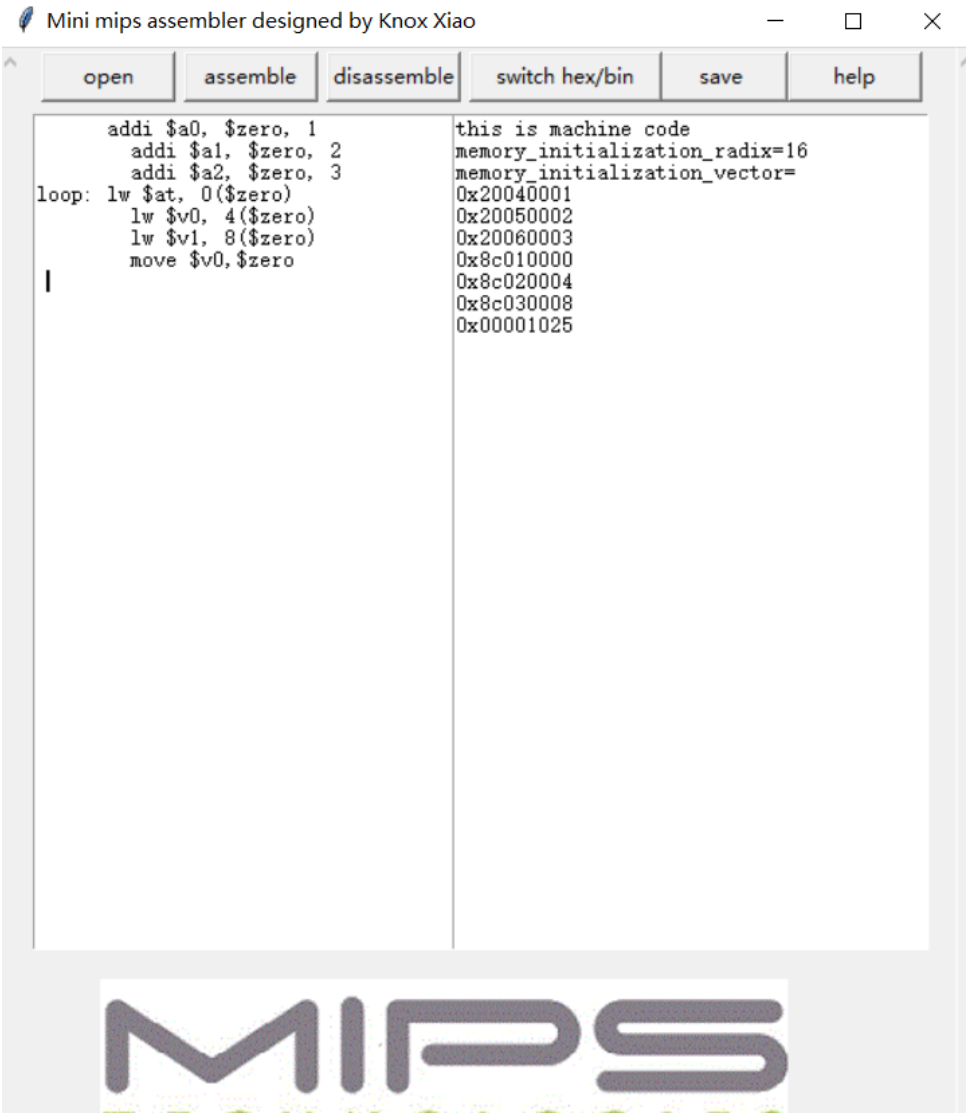
打开后的文件会有右边栏进行显示



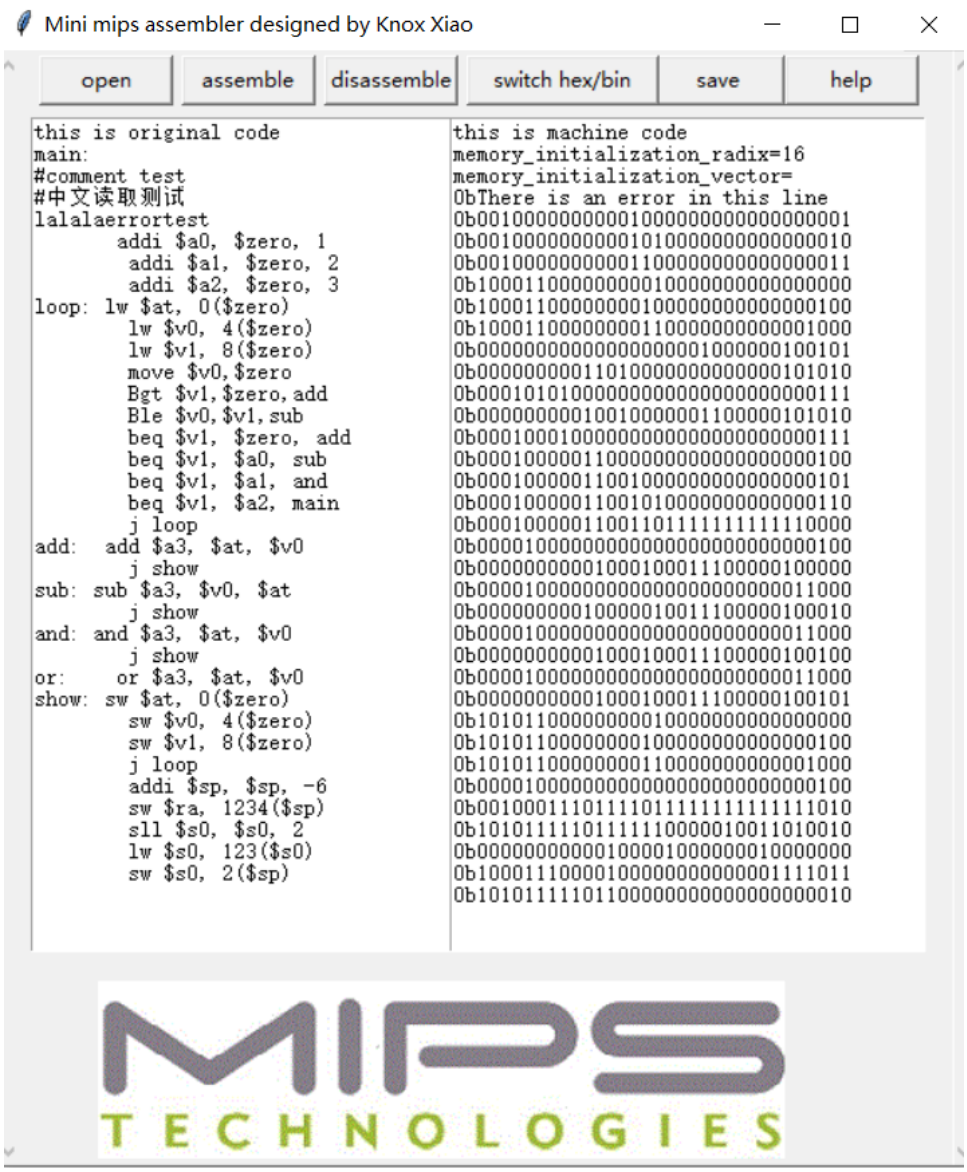
点击assemble可以查看汇编结果



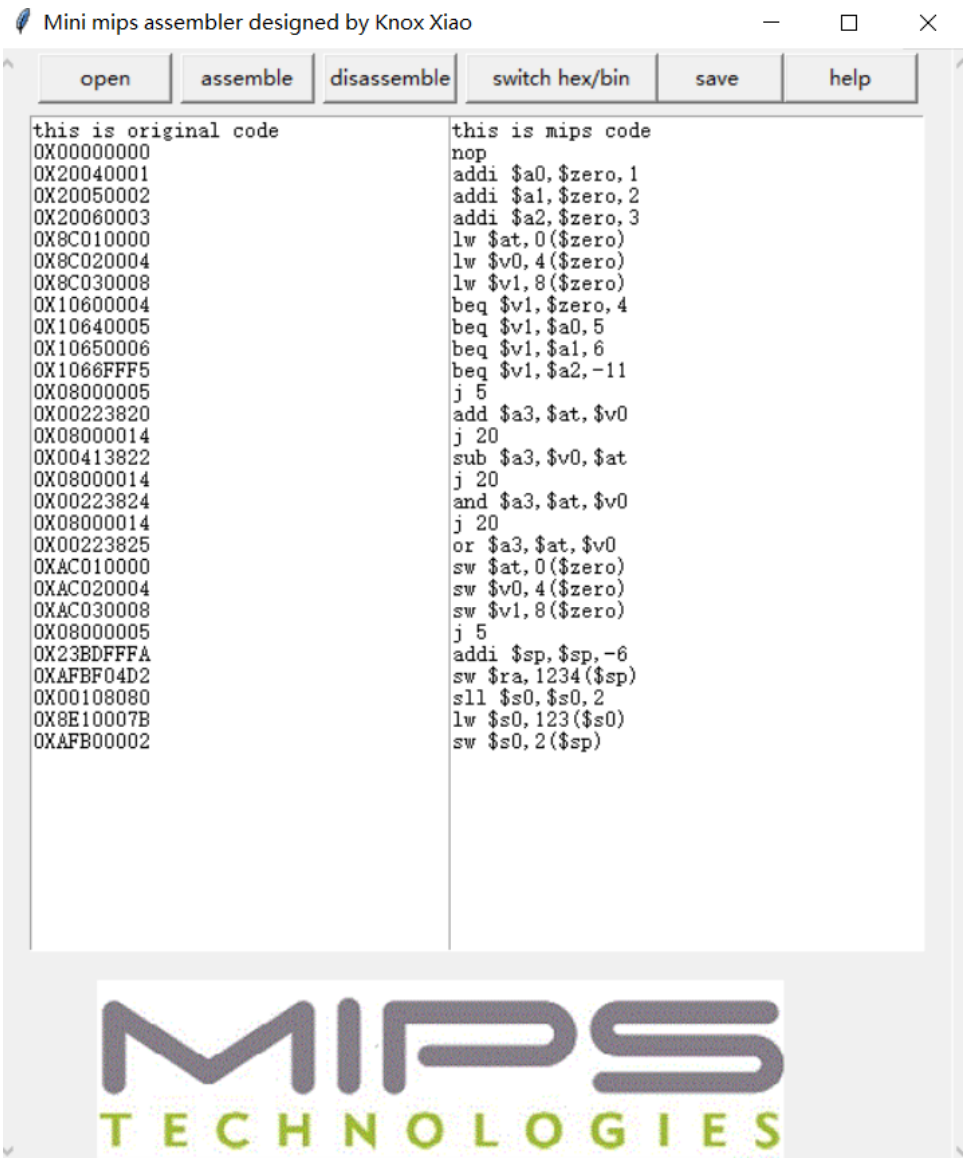
也可以不使用打开文件的形式，而是直接在左边的文本框中进行修改与输入编辑，再次点击assemble可以即时查看结果。



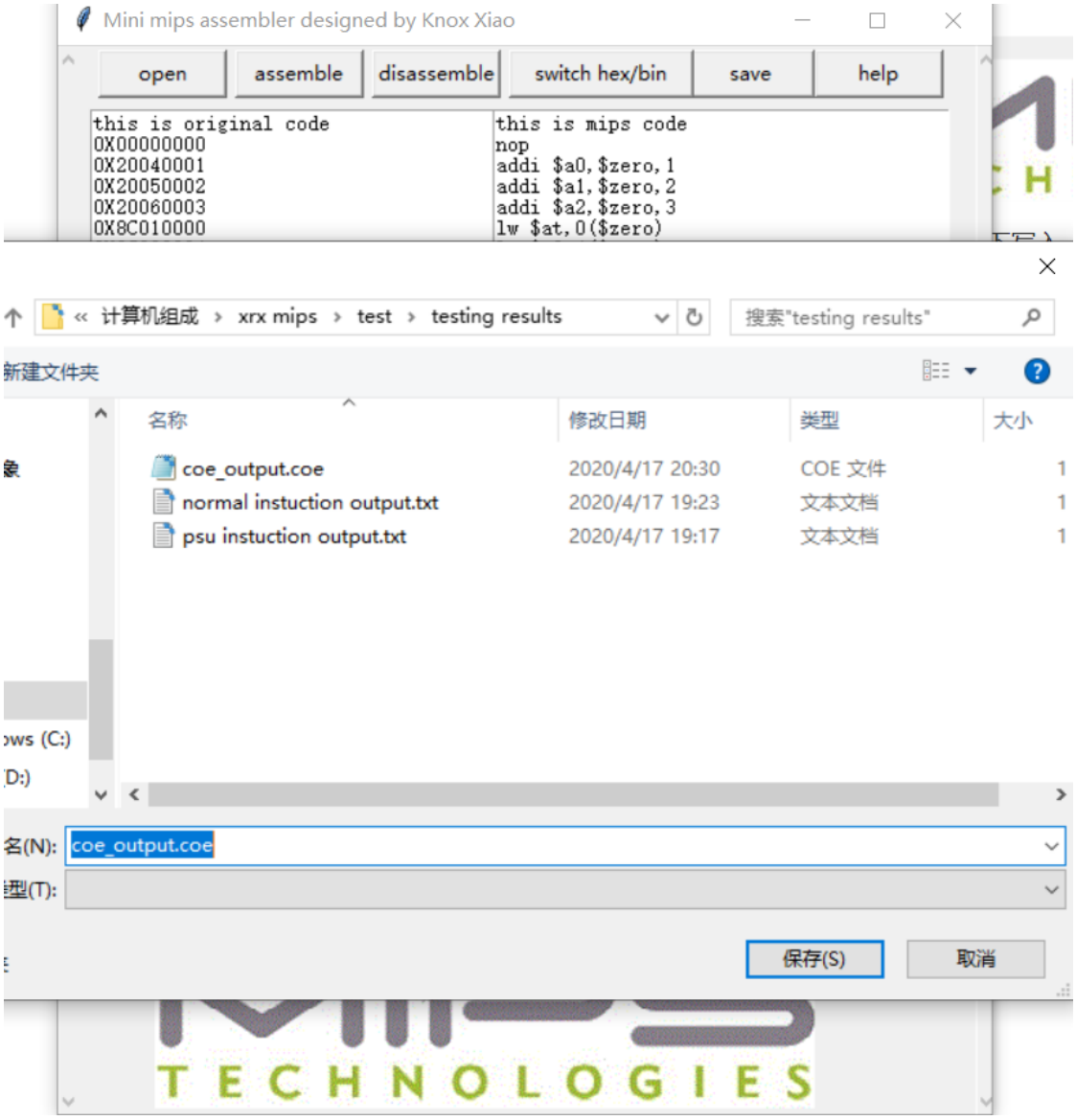
可以点击switch hex/bin按钮进行二进制与十六进制输出的切换



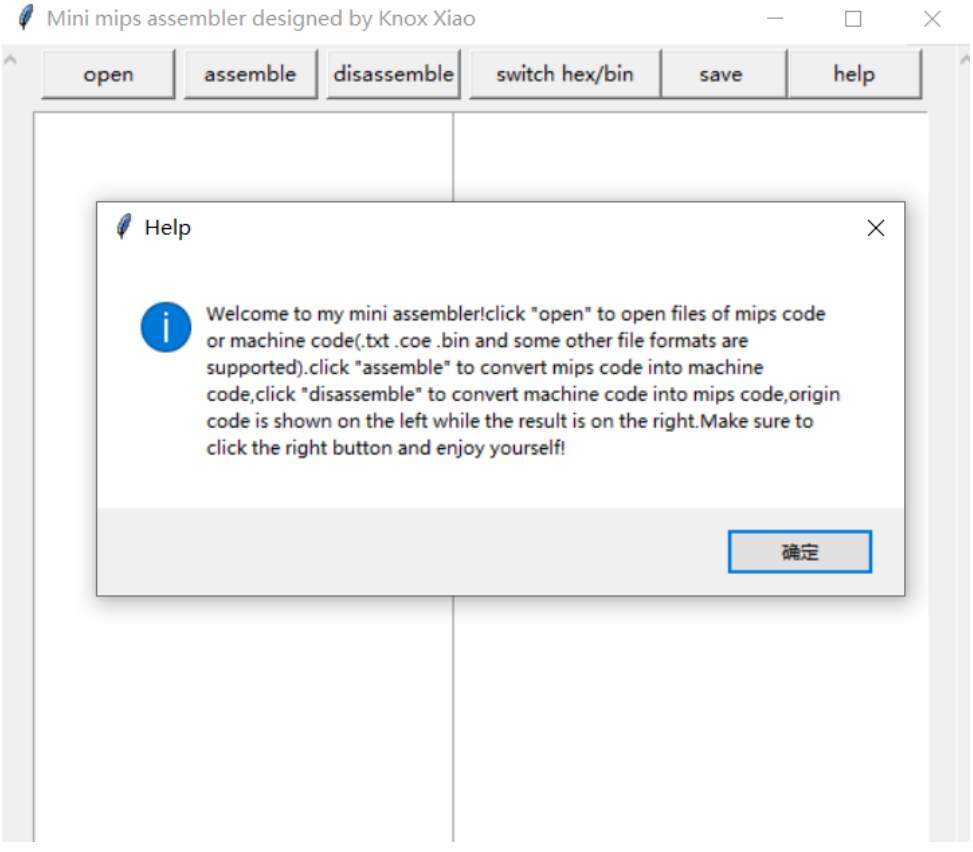
反汇编的过程与汇编过程类似，可以点击open进行打开文件，也可以直接在左边（输入的一直在左边！）进行更改，点击disassemble按钮就可以进行反汇编操作，展示如下图：



程序会自动在当前执行的.py或exe目录下写入一个mipsfrommachine或者machinefrommips.txt的文件（如果有这样的权限的话），当然我们也可以点击save按钮，保存为我们想要的文件与对应的格式（支持.txt .coe .bin）等格式，但文件的后缀需要自己加上！



如果在应用程序的过程中遇到操作问题，可以点击help按钮查看基本的操作指南。



三、汇编器的实现过程与步骤

1.程序的后端的基本逻辑程序

(1)汇编

汇编主体过程是在经过一定的数据处理过程后进行的，在处理的时候需要进行两次扫描，第一次扫描记录下有的

label(如main:的函数头)并记录位置，第二次扫描在碰到这样的label的时候直接替换成对应的位置即可，在进行预处理的时候需要先对注释#进行处理，然后对每条指令按照"," " " 符号分开，依次存储进lists列表就可以，然后按照不同指令的模式对不同指令进行分类（俺大类是R I J类 然后再在之下分出几个小类），对于不同的模式编写对应的处理函数转换为机器码。

预处理的过程如下：

```
    for instruction in newinstructions:
        instruction=instruction.strip()
        if instruction=="":
            continue
        if instruction[0] == '#':
            continue
        if instruction.find(':') != -1:
            cut = re.split(r'[:|#]', instruction)
            position.append(i)
            ch.append(cut[0])
            if(cut[1].strip()!=""):
                instruction = cut[1]
            else:
                instruction=""
    for mips in lists:
            for t in range(1, len(mips)):
                try:
                    index = ch.index(mips[t])
                    if mips[0] in j_1:
                        mips[t] = position[index]
                    else:
                        mips[t] = position[index]-j-1
                except ValueError:
                    continue
            try:
                if mips[0] in r_1:
                    result2 = transfer_r_type1(mips)
                elif mips[0] in r_2:
                    result2 = transfer_r_type2(mips)
                elif mips[0] in i_1:
                    result2 = transfer_i_type1(mips)
                elif mips[0] in i_2:
                    result2 = transfer_i_type2(mips)
                elif mips[0] in i_3:
                    result2 = transfer_i_type3(mips)
                elif mips[0] in j_1:
                    result2 = transfer_j_type(mips)
                else:
                    result2 = transfer_special_type(mips)
```

```
                Result2 = '0b' + result2
```

对于R型中的部分指令转化如下：

```
def transfer_r_type1(mips):
    result = '000000' + reg_code[mips[2]] + reg_code[mips[3]] + reg_code[
        mips[1]] + '00000' + r_type1[mips[0]]
    return result
```

(2)反汇编

反汇编的过程与汇编类似,先进行预处理，去掉无效字段，去掉对机器码先辨别，分为R型与非R型指令，再在这个的基础上拆分为31-26、25-21、20-16、15-10、10-6、5-0几个字段，按照不同种类指令的不同点进行匹配，对不同类的指令进行不同的函数处理即可。

对于R型中的部分指令转化如下：

```
def transfer_r_type1_re(line):
    #                                       rd
rs                      rt
    result=r_type1_re[line[26:32]]+"
"+reg_code_re[line[16:21]]+','+reg_code_re[line[6:11]]+','+reg_code_re[line[11:1
6]]
    return result
```

(3)伪指令的处理

对于这几条伪指令，我进行的操作是在读取的时候检测伪指令，把每条伪指令转化为对应的一条或几条可以操作的基本指令，然后后续对这样的基本指令进行正常的汇编操作即可。

```
        if(instruction[0:5]=="move "):
            instruction="or "+instruction[5:]+",$zero"
        if(instruction[0:4]=="not "):
            instruction="nor "+instruction[4:]+",$zero"
        if(instruction[0:6]=="clear "):
            instruction="add "+instruction[6:]+",$zero,$zero"
        if(instruction[0:4]=="Bgt "or instruction[0:4]=="bgt "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[1]+","+newcut[0]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="bne $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Bge " or instruction[0:4]=="bge "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[0]+","+newcut[1]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="beq $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Blt " or instruction[0:4]=="blt "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[0]+","+newcut[1]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="bne $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Ble "or instruction[0:4]=="ble "):
            newcut=instruction[4:].split(',')
```

```
            instruction="slt "+newcut[1]+","+newcut[0]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="beq $t0,$zero,"+newcut[2]
```

2.程序的可视化操作

本汇编器中采用了python中的tkinter进行ui用户交互界面的设计，虽然tkinter相对于QT有些功能仍然难以实现，但在简单的汇编器中需要的图形化界面用tkinter就可以进行实现。

打开exe时**入场动画的框架**

```
global window
    window = Tk()
    window.geometry('533x331')
    window.title('Welcome to my mini mips assembler')
    photo1 = PhotoImage(file=os.path.split(os.path.realpath(__file__))
[0]+'\source\mips.gif')
    global label_img
    label_img = tk.Label(window, image = photo1)
    label_img.place(relx=0.0, rely=0.0)
    label_img.pack(side=tk.RIGHT)
    global textbegin
    textbegin = Text(window,width=100,height=1)
    textbegin.pack()
    textbegin.place(relx=0.0, rely=0.95)
    textbegin.insert(INSERT,"initializing......")
    label_img.after(1200, destroybegin)
    window.mainloop()
```

创建两个text窗口与滚动条scrollbar，进行关联

```
    global sc1
    sc1=tk.Scrollbar(window,width=10)
    sc1.set(0.2,0)
    sc1.pack(side=tk.LEFT,fill=tk.Y)
    global sc2
    sc2=tk.Scrollbar(window,width=10)
    sc2.set(0.2,0)
    sc2.pack(side=tk.RIGHT,fill=tk.Y)
    global text
    text =Text(window, width=40,height=38,yscrollcommand=sc1.set)
    global text2
    text2 =Text(window, width=40,height=38,yscrollcommand=sc2.set)
    # 两个控件关联
    text.place(x=20,y=40)
    text2.place(x=270,y=40)
```

基本的按钮控件

```
   global b1
    b1 = tk.Button(window, text='open', width=10,height=1,
command=lambda:openmytxt(readjudge))
    b1.place(x=25,y=3)
    global b2
```

```
    b2 = tk.Button(window, text='assemble', width=10,height=1,
command=lambda:showtranslatemachine(writejudge))
    b2.place(x=110,y=3)
    global b3
    b3 = tk.Button(window, text='disassemble', width=10,height=1,
command=lambda:showtranslatemips(writejudge))
    b3.place(x=195,y=3)
    global b4
    b4 = tk.Button(window, text='help', width=10,height=1,
command=showhelpmessage)
    b4.place(x=470,y=3)
    global b5
    b5 = tk.Button(window, text='save', width=10,height=1, command=save_file)
    b5.place(x=390,y=3)
    global b6
    b6 = tk.Button(window, text='switch hex/bin', width=15,height=1,
command=hexchange)
    b6.place(x=280,y=3)
    window.title('Mini mips assembler designed by Knox Xiao')
```

其中每个按钮关联着他们对应的回调函数，负责对显示的text内容进行调整。

`filedialogue`的文件读取与保存可视化框架

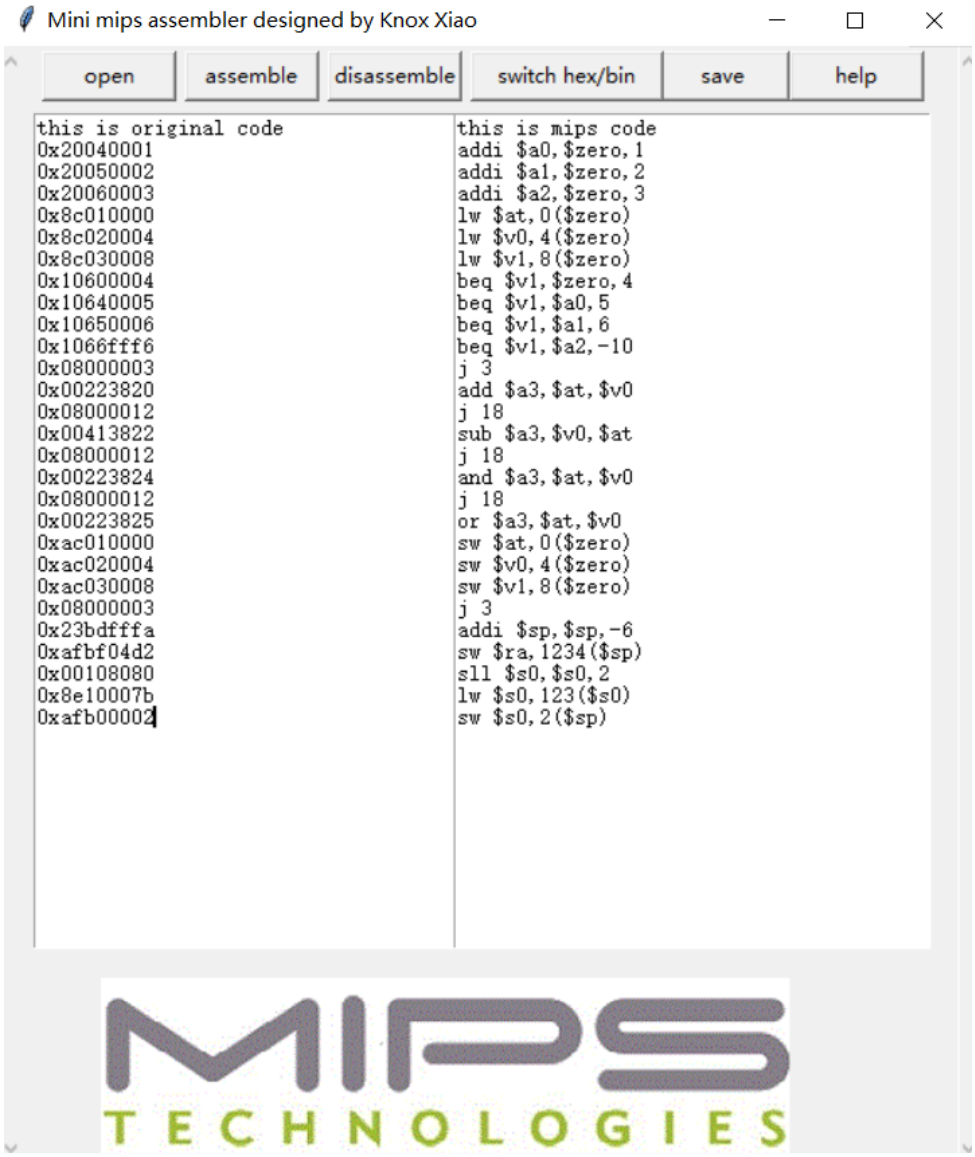
```
    global file_path
    file_path= filedialog.askopenfilename(title=u'选择文件', initialdir=
(os.path.expanduser(default_dir)))
    #f1 = open(path + '\\mips.txt')
    f1=open(file_path,encoding='utf-8')
```

四、汇编器的简单测试

本实验的所有的测试数据都在一同上交的文件夹中的testing data中有所展示

(1)简单的汇编与反汇编的对照测试

对之前的结果进行反汇编：


Mini mips assembler designed by Knox Xiao
open
assemble
disassemble
switch hex/bin
save
help
this is original code
0x20040001
0x20050002
0x20060003
0x8c010000
0x8c020004
0x8c030008
0x10600004
0x10640005
0x10650006
0x1066fff6
0x08000003
0x00223820
0x08000012
0x00413822
0x08000012
0x00223824
0x08000012
0x00223825
0xac010000
0xac020004
0xac030008
0x08000003
0x23bdfffa
0xafbf04d2
0x00108080
0x8e10007b
0xafb00002
this is mips code
addi $a0,$zero,1
addi $a1,$zero,2
addi $a2,$zero,3
lw $at,0($zero)
lw $v0,4($zero)
lw $v1,8($zero)
beq $v1,$zero,4
beq $v1,$a0,5
beq $v1,$a1,6
beq $v1,$a2,-10
j 3
add $a3,$at,$v0
j 18
sub $a3,$v0,$at
j 18
and $a3,$at,$v0
j 18
or $a3,$at,$v0
sw $at,0($zero)
sw $v0,4($zero)
sw $v1,8($zero)
j 3
addi $sp,$sp,-6
sw $ra,1234($sp)
sll $s0,$s0,2
lw $s0,123($s0)
sw $s0,2($sp)
MIPS
TECHNOLOGIES

(2)对伪代码进行测试

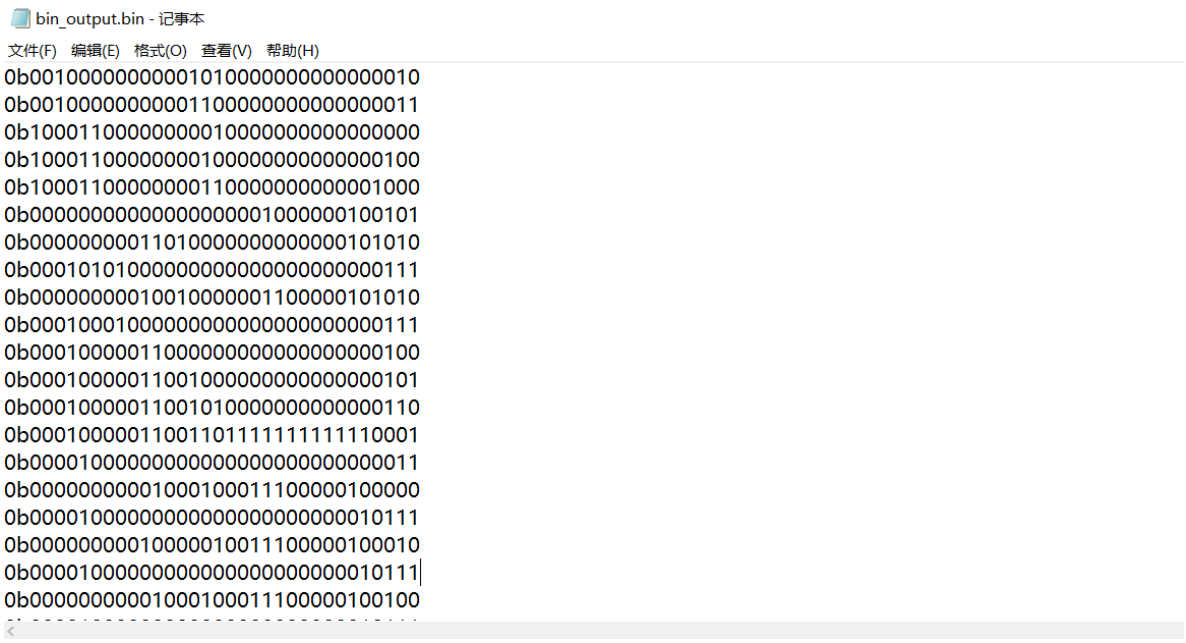
(3)对于保存文件的测试

保存为coe_output.coe,文件内容如下

```
memory_initialization_radix=16
memory_initialization_vector=
0x20040001
0x20050002
0x20060003
0x8c010000
0x8c020004
0x8c030008
0x00001025
0x0068002a
0x15000007
0x0048182a
0x11000007
0x10600004
0x10640005
0x10650006
0x1066fff1
0x08000003
0x00223820
0x08000017
0x00413822
```

```
0x08000017
0x00223824
0x08000017
0x00223825
0xac010000
0xac020004
0xac030008
0x08000003
0x23bdfffa
0xafbf04d2
0x00108080
0x8e10007b
0xafb00002
```

保存为bin_output.coe,文件内容如下：



五、实验心得与体会

这次汇编器的编写是一次一边学习一边编写的过程，在学习与调试的过程中花费了很多的时间，但也获得了非常多的收获，通过这次实验，我对MIPS指令的操作更加熟悉了，对于python编程的流程与简单的tkinter的ui界面设计也更加熟悉，但是因为时间的原因，对于extension2中的debug模式暂时还没有能够完整实现，希望之后可以在现有的基础上实现debug功能。

附录：程序源代码

```
import re
import os,sys,stat
path = os.getcwd()
import tkinter as tk
from tkinter import *
from tkinter import filedialog,dialog
from tkinter import scrolledtext
import tkinter.messagebox
import time
```

```
#global readjudge
readjudge=0
#global writejudge
writejudge=0
binorhex=False



all_ch = {
    'add', 'addu', 'sub', 'subu', 'and', 'or', 'xor', 'nor', 'slt', 'sltu',
    'sllv', 'srlv', 'srav', 'addi', 'addiu', 'andi', 'ori', 'xori', 'beq',
    'bne', 'slti', 'sltiu', 'j', 'jal', 'sll', 'srl', 'sra', 'jr', 'lw', 'sw',
    'lui', '$zero', '$at', '$v0', '$v1', '$a0', '$a1', '$a2', '$a3', '$t0',
    '$t1', '$t2', '$t3', '$t4', '$t5', '$t6', '$t7', '$s0', '$s1', '$s2',
    '$s3', '$s4', '$s5', '$s6', '$s7', '$t8', '$t9', '$k0', '$k1', '$gp',
    '$sp', '$fp', '$ra'
}
r_1 = {
    'add', 'addu', 'sub', 'subu', 'and', 'or', 'xor', 'nor', 'slt', 'sltu',
    'sllv', 'srlv', 'srav'
}
r_2 = {'sll', 'srl', 'sra'}
i_1 = {'addi', 'addiu', 'andi', 'ori', 'xori', 'slti', 'sltiu'}
i_2 = {'lhx', 'lh', 'lwx', 'lw', 'lhux', 'lhu', 'shx', 'sh', 'swx', 'sw'}
i_3 = {'beq', 'bne'}
j_1 = {'j', 'jal'}

r_type1 = {
    'add': '100000',
    'addu': '100001',
    'sub': '100010',
    'subu': '100011',
    'and': '100100',
    'or': '100101',
    'xor': '100110',
    'nor': '100111',
    'slt': '101010',
    'sltu':
    '101011',  # It's an I-type instruction with the same form as r_type1.
    'sllv': '000100',
    'srlv': '000110',
    'srav': '000111'
}

r_type2 = {'sll': '000000', 'srl': '000010', 'sra': '000011'}

i_type1 = {
    'addi': '001000',
    'addiu': '001001',
    'andi': '001100',
    'ori': '001101',
    'xori': '001110',
    'slti':
    '001010',  # It's an R-type instruction with the same form as i_type1.
    'sltiu': '001011'
}

i_type2 = {
```

```
    'lhx': '100000',
    'lh': '100001',
    'lwx': '100010',
    'lw': '100011',
    'lhux': '100100',
    'lhu': '100101',
    'shx': '101000',
    'sh': '101001',
    'swx': '101010',
    'sw': '101011'
}

i_type3 = {
    'beq': '000100',
    'bne': '000101',
}

j_type = {'j': '000010', 'jal': '000011'}

reg_code = {
    '$zero': '00000',
    '$at': '00001',
    '$v0': '00010',
    '$v1': '00011',
    '$a0': '00100',
    '$a1': '00101',
    '$a2': '00110',
    '$a3': '00111',
    '$t0': '01000',
    '$t1': '01001',
    '$t2': '01010',
    '$t3': '01011',
    '$t4': '01100',
    '$t5': '01101',
    '$t6': '01110',
    '$t7': '01111',
    '$s0': '10000',
    '$s1': '10001',
    '$s2': '10010',
    '$s3': '10011',
    '$s4': '10100',
    '$s5': '10101',
    '$s6': '10110',
    '$s7': '10111',
    '$t8': '11000',
    '$t9': '11001',
    '$k0': '11010',
    '$k1': '11011',
    '$gp': '11100',
    '$sp': '11101',
    '$fp': '11110',
    '$ra': '11111'
}


r_type1_re = {'100000': 'add',
 '100001': 'addu',
 '100010': 'sub',
```

```
 '100011': 'subu',
 '100100': 'and',
 '100101': 'or',
 '100110': 'xor',
 '100111': 'nor',
 '101010': 'slt',
 '101011': 'sltu',
 '000100': 'sllv',
 '000110': 'srlv',
 '000111': 'srav'}

r_type2_re = {'000000': 'sll', '000010': 'srl', '000011': 'sra'}

i_type1_re = {'001000': 'addi',
    '001001': 'addiu',
    '001100': 'andi',
    '001101': 'ori',
    '001110': 'xori',
    '001010': 'slti',
    '001011': 'sltiu'}

i_type2_re = {'100000': 'lhx',
    '100001': 'lh',
    '100010': 'lwx',
    '100011': 'lw',
    '100100': 'lhux',
    '100101': 'lhu',
    '101000': 'shx',
    '101001': 'sh',
    '101010': 'swx',
    '101011': 'sw'}

i_type3_re = {'000100': 'beq', '000101': 'bne'}

j_type_re = {'000010': 'j', '000011': 'jal'}

reg_code_re = {'00000': '$zero',
    '00001': '$at',
    '00010': '$v0',
    '00011': '$v1',
    '00100': '$a0',
    '00101': '$a1',
    '00110': '$a2',
    '00111': '$a3',
    '01000': '$t0',
    '01001': '$t1',
    '01010': '$t2',
    '01011': '$t3',
    '01100': '$t4',
    '01101': '$t5',
    '01110': '$t6',
    '01111': '$t7',
    '10000': '$s0',
    '10001': '$s1',
    '10010': '$s2',
    '10011': '$s3',
    '10100': '$s4',
    '10101': '$s5',
```

```
    '10110': '$s6',
    '10111': '$s7',
    '11000': '$t8',
    '11001': '$t9',
    '11010': '$k0',
    '11011': '$k1',
    '11100': '$gp',
    '11101': '$sp',
    '11110': '$fp',
    '11111': '$ra'}

def showhelpmessage():
    tk.messagebox.showinfo(title='Help', message='Welcome to my mini
assembler!click "open" to open files of mips code or \
machine code(.txt .coe .bin and some other file formats are supported).click
"assemble" to convert mips code into machine code,click "disassemble" to convert
machine code into mips \
code,origin code is shown on the left while the result is on the right.Make sure
to click the right button and enjoy yourself!')

def openmytxt(vara):
    if(vara==1):
        text.delete('1.0','end')
    global readjudge
    readjudge=1
    default_dir = r"文件路径"
    global file_path
    file_path= filedialog.askopenfilename(title=u'选择文件', initialdir=
(os.path.expanduser(default_dir)))
    #f1 = open(path + '\\mips.txt')
    f1=open(file_path,encoding='utf-8')
    global instructions
    instructions = f1.readlines()
    text.insert(INSERT, "this is original code\n")
    with open(file_path,encoding='utf-8') as f111:
        for each_line in f111:
            text.insert(INSERT, each_line)
    text.insert(INSERT, "\n")


def showtranslatemips(varb):
    if(varb==1):
        text2.delete('1.0','end')
    global writejudge
    writejudge=1
    global lines
    lines = []
    global lists
    lists = []
    global position
    position = []
    global ch
    ch = []
    i = 0
    newinstructions=(text.get("1.0","end")).split("\n")
    if (newinstructions[0]=="this is original code"):
        newinstructions.pop(0)
    newinstructions.pop()
```

```
    for instruction in newinstructions:
        try:
            instruction=instruction.strip()
            if instruction=="":
                continue
            instruction=instruction[2:]#去除0x
            instruction=bin(int(instruction,16))[2:]
            instruction=instruction.zfill(32)
            lines.append(instruction.strip())
            i = i + 1
        except ValueError:
            instruction="11111111111111111111111111111111"
    j = 0
    text2.insert(INSERT, "this is mips code\n")
    os.chmod(os.path.split(os.path.realpath(__file__))[0],stat.S_IRWXO)
    f2 = open(os.path.split(os.path.realpath(__file__))
[0]+"\mipsfrommachine.txt", 'w')
    for line in lines:
        if (line=="00000000000000000000000000000000"):
            result3="nop"
        elif (line[0:6]) in i_type1_re.keys():
            result3=transfer_i_type1_re(line)
        elif (line[0:6]) in i_type2_re.keys():
            result3=transfer_i_type2_re(line)
        elif (line[0:6]) in i_type3_re.keys():
            result3=transfer_i_type3_re(line)
        elif (line[0:6]) in j_type_re.keys():
            result3=transfer_j_type_re(line)
        elif line[0:6]=="000000":
            if (line[26:32]) in r_type1_re.keys():
                result3=transfer_r_type1_re(line)
            elif(line[26:32]) in r_type2_re.keys():
                result3=transfer_r_type2_re(line)
            else:
                result3=transfer_special_r_type_re(line)
        else:
            result3 ="there is error in this line!!!"
        #Result2 = '0b' + result2
        print(result3)
        f2.write(result3+"\n")
        text2.insert(INSERT, result3+'\n')
        j = j + 1
    f2.close()

def showtranslatemachine(varb):
    if(varb==1):
        text2.delete('1.0','end')
    global writejudge
    writejudge=1
    global lines
    lines = []
    global lists
    lists = []
    global position
    position = []
    global ch
    ch = []
    i = 0
```

```
    newinstructions=(text.get("1.0","end")).split("\n")
    if (newinstructions[0]=="this is original code"):
        newinstructions.pop(0)
    newinstructions.pop()
    for instruction in newinstructions:
        instruction=instruction.strip()
        if instruction=="":
            continue
        if instruction[0] == '#':
            continue
        if instruction.find(':') != -1:
            cut = re.split(r'[:|#]', instruction)
            position.append(i)
            ch.append(cut[0])
            if(cut[1].strip()!=""):
                instruction = cut[1]
            else:
                instruction=""
        if(instruction[0:5]=="move "):
            instruction="or "+instruction[5:]+",$zero"
        if(instruction[0:4]=="not "):
            instruction="nor "+instruction[4:]+",$zero"
        if(instruction[0:6]=="clear "):
            instruction="add "+instruction[6:]+",$zero,$zero"
        if(instruction[0:4]=="Bgt "or instruction[0:4]=="bgt "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[1]+","+newcut[0]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="bne $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Bge " or instruction[0:4]=="bge "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[0]+","+newcut[1]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="beq $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Blt " or instruction[0:4]=="blt "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[0]+","+newcut[1]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="bne $t0,$zero,"+newcut[2]
        if(instruction[0:4]=="Ble "or instruction[0:4]=="ble "):
            newcut=instruction[4:].split(',')
            instruction="slt "+newcut[1]+","+newcut[0]+","+"$t0"
            lines.append(instruction.lstrip())
            i = i + 1
            instruction="beq $t0,$zero,"+newcut[2]
        if(instruction.strip()!=""):
            lines.append(instruction.lstrip())
            i = i + 1
    for line in lines:
        tmp = re.split(' |,|\n|\(|\)', line)
        result1 = [x.strip() for x in tmp if x.strip() != '']
        lists.append(result1)
    j = 0
    text2.insert(INSERT, "this is machine code\n")
    text2.insert(INSERT, "memory_initialization_radix=16\n")
```

```
    text2.insert(INSERT, "memory_initialization_vector=\n")
    os.chmod(os.path.split(os.path.realpath(__file__))[0],stat.S_IRWXO)
    with open(os.path.split(os.path.realpath(__file__))
[0]+"\machinecodefrommips.txt", 'w') as f2:
        for mips in lists:
            for t in range(1, len(mips)):
                try:
                    index = ch.index(mips[t])
                    if mips[0] in j_1:
                        mips[t] = position[index]
                    else:
                        mips[t] = position[index]-j-1
                except ValueError:
                    continue
            try:
                if mips[0] in r_1:
                    result2 = transfer_r_type1(mips)
                elif mips[0] in r_2:
                    result2 = transfer_r_type2(mips)
                elif mips[0] in i_1:
                    result2 = transfer_i_type1(mips)
                elif mips[0] in i_2:
                    result2 = transfer_i_type2(mips)
                elif mips[0] in i_3:
                    result2 = transfer_i_type3(mips)
                elif mips[0] in j_1:
                    result2 = transfer_j_type(mips)
                else:
                    result2 = transfer_special_type(mips)
                Result2 = '0b' + result2
                if(binorhex):
                    print(Result2)
                    f2.write(Result2 + '\n')
                    text2.insert(INSERT, Result2 + '\n')
                else:
                    print("{:#010x}".format((int(Result2, 2))))
                    f2.write("{:#010x}".format((int(Result2, 2))) + '\n')
                    text2.insert(INSERT, "{:#010x}".format((int(Result2, 2)))+
'\n')
            except ValueError:
                print("Error this line")
                f2.write("Error this line" + '\n')
                text2.insert(INSERT, "Error this line" + '\n')
            j = j + 1
    f2.close()



def binary(num, bit):
    return (bin(((1 << bit) - 1) & num)[2:]).zfill(bit)


def transfer_r_type1_re(line):
    #                                       rd
rs                      rt
    result=r_type1_re[line[26:32]]+"
"+reg_code_re[line[16:21]]+','+reg_code_re[line[6:11]]+','+reg_code_re[line[11:1
6]]
```

```
    return result


def transfer_r_type2_re(line):
    result=r_type2_re[line[26:32]]+"
"+reg_code_re[line[16:21]]+','+reg_code_re[line[11:16]]+','+str(int((line[21:26]
),2))
    return result



def transfer_i_type1_re(line):
    if(line[16]=="1"):
        result=i_type1_re[line[0:6]]+"
"+reg_code_re[line[11:16]]+','+reg_code_re[line[6:11]]+',-'+str(65536-
int((line[16:32]),2))
    else:
        result=i_type1_re[line[0:6]]+"
"+reg_code_re[line[11:16]]+','+reg_code_re[line[6:11]]+','+str(int((line[16:32])
,2))
    return result


def transfer_i_type2_re(line):
    result=i_type2_re[line[0:6]]+"
"+reg_code_re[line[11:16]]+','+str(int((line[16:32]),2)) +"
("+reg_code_re[line[6:11]]+')'
    return result

def transfer_i_type3_re(line):
    if(line[16]=="1"):
        result=i_type3_re[line[0:6]]+"
"+reg_code_re[line[6:11]]+','+reg_code_re[line[11:16]]+',-'+str(65536-
int((line[16:32]),2))
    else:
        result=i_type3_re[line[0:6]]+"
"+reg_code_re[line[6:11]]+','+reg_code_re[line[11:16]]+','+str(int((line[16:32])
,2))
    return result


def transfer_j_type_re(line):
    result=j_type_re[line[0:6]]+" "+str(int((line[6:32]),2))
    return result


def transfer_special_r_type_re(line):
    result="jr"+ reg_code_re[line[6:11]]


def binary(num, bit):
    return (bin(((1 << bit) - 1) & num)[2:]).zfill(bit)


def transfer_r_type1(mips):
    result = '000000' + reg_code[mips[2]] + reg_code[mips[3]] + reg_code[
        mips[1]] + '00000' + r_type1[mips[0]]
    return result
```

```
def transfer_r_type2(mips):
    result = '00000000000' + reg_code[mips[2]]  + reg_code[
        mips[1]] + binary(int(mips[3]), 5) + r_type2[mips[0]]
    return result


def transfer_i_type1(mips):
    result = i_type1[mips[0]] + reg_code[mips[2]] + reg_code[mips[1]] + binary(
        int(mips[3]), 16)
    return result


def transfer_i_type2(mips):
    result = i_type2[mips[0]] + reg_code[mips[3]] + reg_code[mips[1]] + binary(
        int(mips[2]), 16)
    return result


def transfer_i_type3(mips):
    result = i_type3[mips[0]] + reg_code[mips[1]] + reg_code[mips[2]] + binary(
        int(mips[3]), 16)
    return result


def transfer_j_type(mips):
    tmp = int(mips[1])
    result = j_type[mips[0]] + binary(tmp, 26)
    return result


def transfer_special_type(mips):
    result="There is an error in this line"#'00000000000000000000000000000000'
    if mips[0] == 'jr':
        result = '000000' + reg_code[
            mips[1]] + '00000' + '00000' + '00000' + '001000'
    elif mips[0] == 'lui':
        result = '001111' + '00000' + reg_code[mips[1]] + binary(
            int(mips[2]), 16)
    elif mips[0] == 'bgezal':
        result = '000001' + reg_code[mips[1]] + '10001' + binary(
            int(mips[2]), 16)
    elif mips[0] == 'jalr':
        result = '000000' + reg_code[mips[1]] + '00000' + reg_code[
            mips[2]] + '00000' + '001001'
    elif mips[0] == 'mfc0':
        result = '010000' + '00000' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '000000'
    elif mips[0] == 'mtc0':
        result = '010000' + '00100' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '000000'
    elif mips[0] == 'eret':
        result = '010000' + '10000' + '00000' + '00000' + '00000' + '011000'
    elif mips[0] == 'syscal':
        result = '000000' + '00000' + '00000' + '00000' + '00000' + '001100'
    elif mips[0] == 'mul':
        result = '011100' + reg_code[mips[2]] + reg_code[mips[3]] + reg_code[
```

```
            mips[1]] + '00000' + '000010'
    elif mips[0] == 'mult':
        result = '000000' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '00000' + '011000'
    elif mips[0] == 'multu':
        result = '000000' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '00000' + '011001'
    elif mips[0] == 'div':
        result = '000000' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '00000' + '011010'
    elif mips[0] == 'divu':
        result = '000000' + reg_code[mips[1]] + reg_code[
            mips[2]] + '00000' + '00000' + '011011'
    elif mips[0] == 'mfhi':
        result = '000000' + '00000' + '00000' + reg_code[
            mips[1]] + '00000' + '00000' + '010000'
    elif mips[0] == 'mflo':
        result = '000000' + '00000' + '00000' + reg_code[
            mips[1]] + '00000' + '00000' + '010010'
    elif mips[0] == 'mthi':
        result = '000000' + reg_code[
            mips[1]] + '00000' + '00000' + '00000' + '010001'
    elif mips[0] == 'mtlo':
        result = '000000' + reg_code[
            mips[1]] + '00000' + '00000' + '00000' + '010011'
    return result

def save_file():
    global file_path
    global file_text
    file_path = filedialog.asksaveasfilename(title=u'save files')
    print('保存文件：', file_path)
    file_text = text2.get('2.0','end')
    if file_path is not None:
        with open(file=file_path, mode='a+', encoding='utf-8') as file:
            file.write(file_text)
        dialog.Dialog(None, {'title': 'File Modified', 'text': '保存完成',
'bitmap': 'warning', 'default': 0,
                'strings': ('OK','cancel')})
        print('保存完成')

def hexchange():
    global binorhex
    binorhex=~binorhex
    showtranslatemachine(writejudge)


def destroybegin():
    global beginindexd
    beginindex=1
    textbegin.destroy()
    label_img.destroy()
    window.geometry('580x665')
    label_img.destroy()
    global sc1
    sc1=tk.Scrollbar(window,width=10)
    sc1.set(0.2,0)
    sc1.pack(side=tk.LEFT,fill=tk.Y)
```

```
    global sc2
    sc2=tk.Scrollbar(window,width=10)
    sc2.set(0.2,0)
    sc2.pack(side=tk.RIGHT,fill=tk.Y)
    global text
    text =Text(window, width=40,height=38,yscrollcommand=sc1.set)
    global text2
    text2 =Text(window, width=40,height=38,yscrollcommand=sc2.set)
    # 两个控件关联
    text.place(x=20,y=40)
    text2.place(x=270,y=40)
    #text.config(yscrollcommand=b1.set)
    global b1
    b1 = tk.Button(window, text='open', width=10,height=1,
command=lambda:openmytxt(readjudge))
    b1.place(x=25,y=3)
    global b2
    b2 = tk.Button(window, text='assemble', width=10,height=1,
command=lambda:showtranslatemachine(writejudge))
    b2.place(x=110,y=3)
    global b3
    b3 = tk.Button(window, text='disassemble', width=10,height=1,
command=lambda:showtranslatemips(writejudge))
    b3.place(x=195,y=3)
    global b4
    b4 = tk.Button(window, text='help', width=10,height=1,
command=showhelpmessage)
    b4.place(x=470,y=3)
    global b5
    b5 = tk.Button(window, text='save', width=10,height=1, command=save_file)
    b5.place(x=390,y=3)
    global b6
    b6 = tk.Button(window, text='switch hex/bin', width=15,height=1,
command=hexchange)
    b6.place(x=280,y=3)
    window.title('Mini mips assembler designed by Knox Xiao')
    sc1.config(command=text.yview)
    sc2.config(command=text2.yview)
    print(sys.path[0])
    photo2 = PhotoImage(file=os.path.split(os.path.realpath(__file__))
[0]+'\source\mips2.gif')
    labe2_img = tk.Label(window, image = photo2)
    labe2_img.pack()
    labe2_img.place(relx=0.1, rely=0.832)
    window.mainloop()

def main():
    beginindex=0
    global finename
    filename = sys.argv[0]
    global dirname
    dirname = os.path.dirname(filename)
    global abspath
    abspath = os.path.abspath(dirname)
    global window
    window = Tk()
    window.geometry('533x331')
    window.title('Welcome to my mini mips assembler')
```

```
    photo1 = PhotoImage(file=os.path.split(os.path.realpath(__file__))
[0]+'\source\mips.gif')
    global label_img
    label_img = tk.Label(window, image = photo1)
    label_img.place(relx=0.0, rely=0.0)
    label_img.pack(side=tk.RIGHT)
    global textbegin
    textbegin = Text(window,width=100,height=1)
    textbegin.pack()
    textbegin.place(relx=0.0, rely=0.95)
    textbegin.insert(INSERT,"initializing......")
    label_img.after(1200, destroybegin)
    window.mainloop()

if __name__ == "__main__":
    myt0=time.process_time()
    main()
```