ZPC 2016
ZHEJIANG UNIVERSITY
浙江大学计算机学院

MIPS 模拟器设计

---自然选择前进四

编号 666666

目录

MIPS 模拟器实验报告

小组基本信息：

组名：自然选择前进四

项目负责经理：钱旭峰

其他组员：沈旭东、王海容、黄一伦

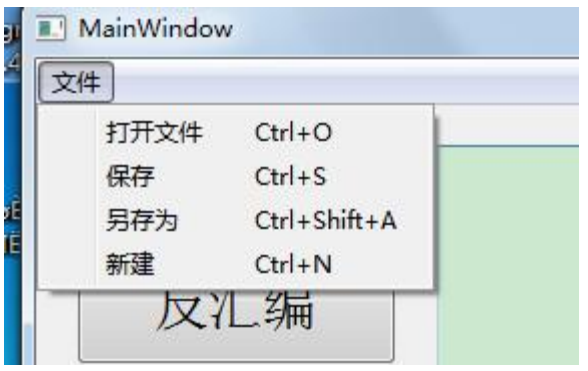
使用手册：

我们采用了界面操作，界面简单易懂，操作方便

如下图：

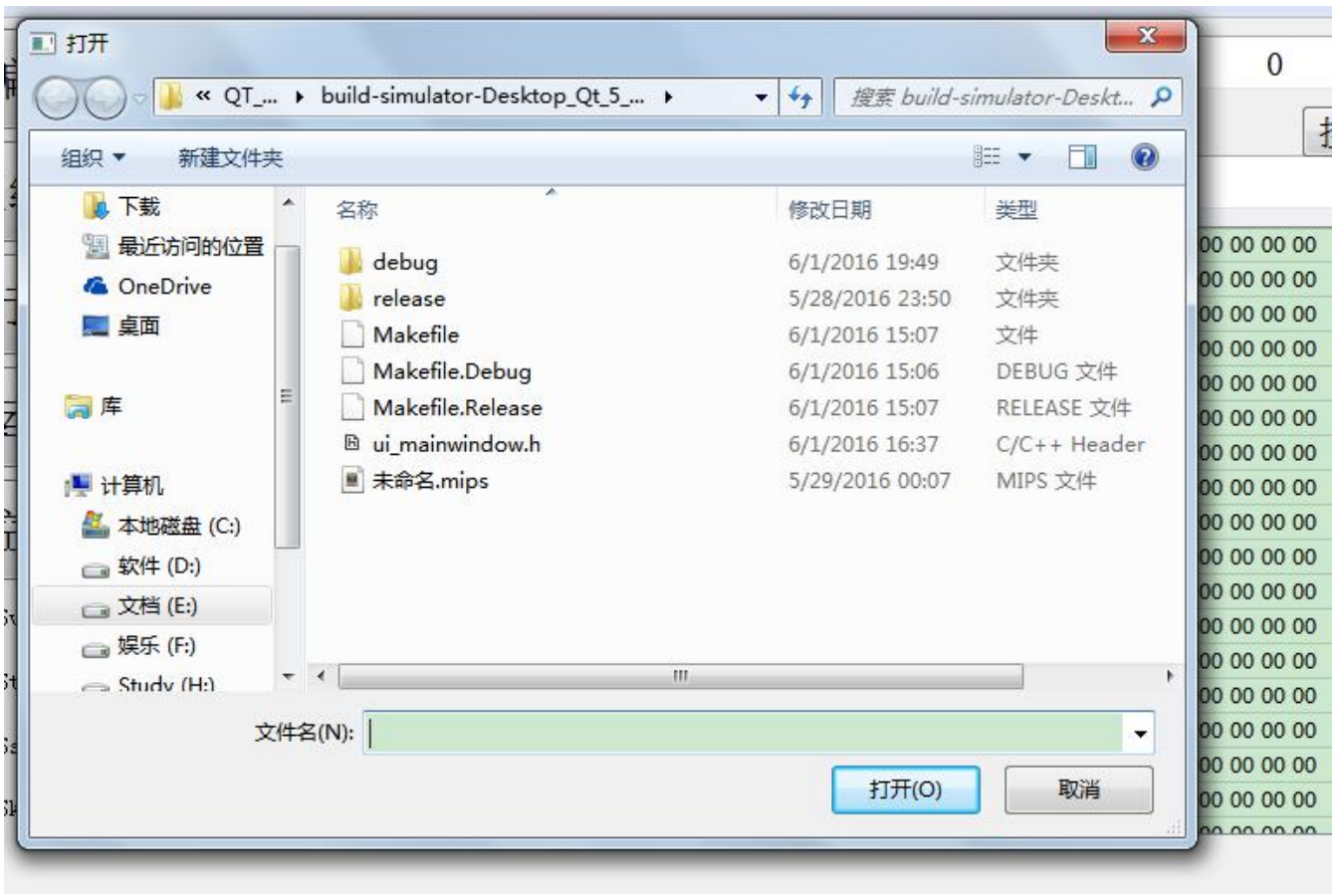
接下来我将为您详细讲解操作细节：

我们的软件支持文件的打开与保存



打开文件：

按打开文件的按钮会跳出如下对话框：



保存文件：

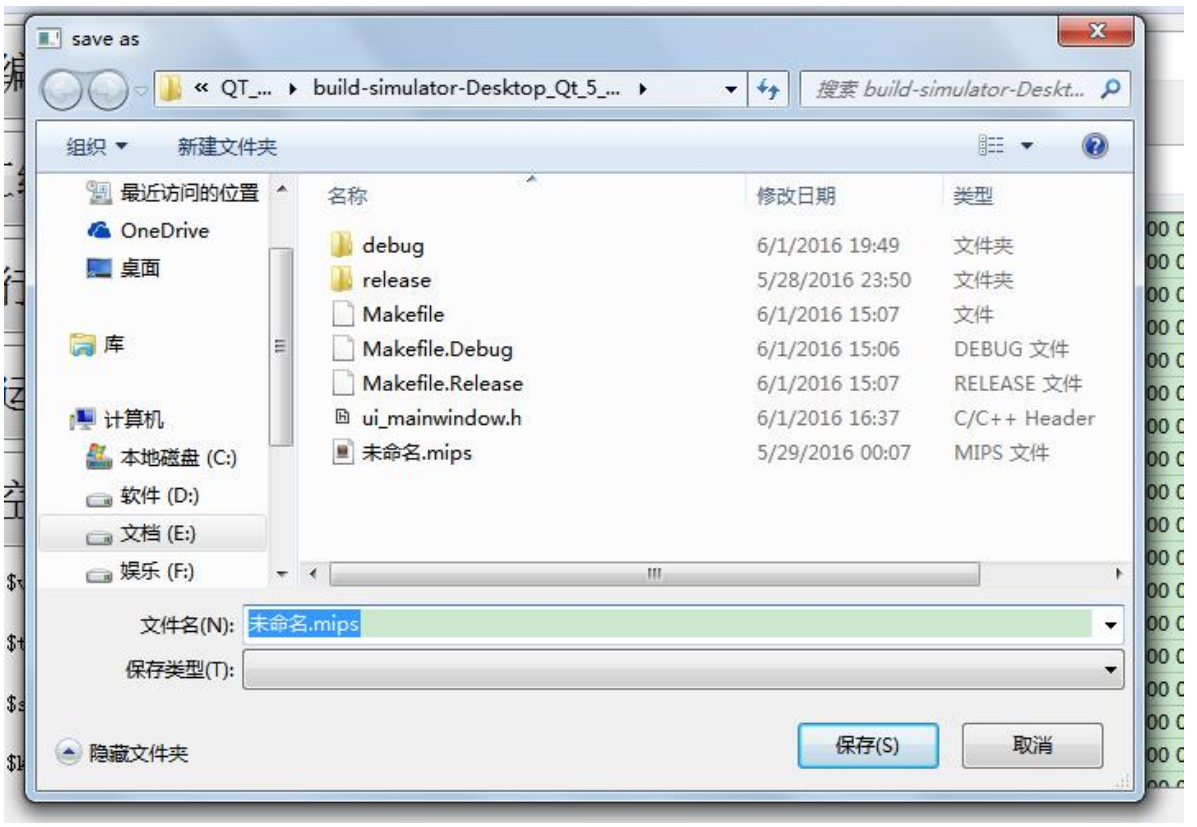
您可以选择保存的地址与文件名进行文件的保存

如果你按下保存按钮，若您曾经将这份文件保存过，则您在文本框的内容会保存到原地址

若您没有曾经将这份文件保存过，则系统将自动调用另存为函数

效果如另存为

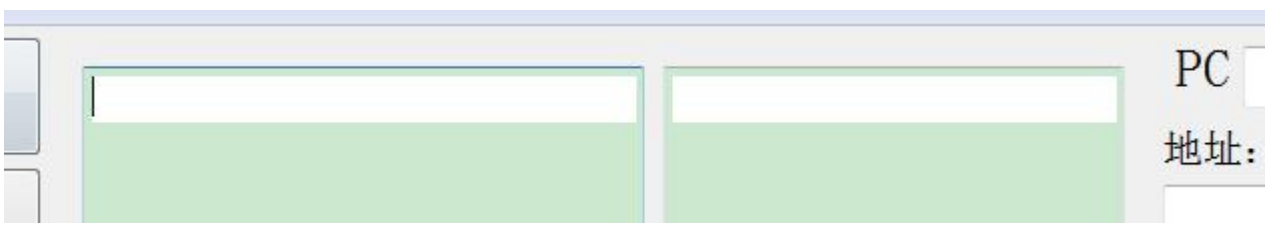
当您按下另存为按钮，软件会弹出这样的窗口：



您可以选择保存的地址与文件名进行保存

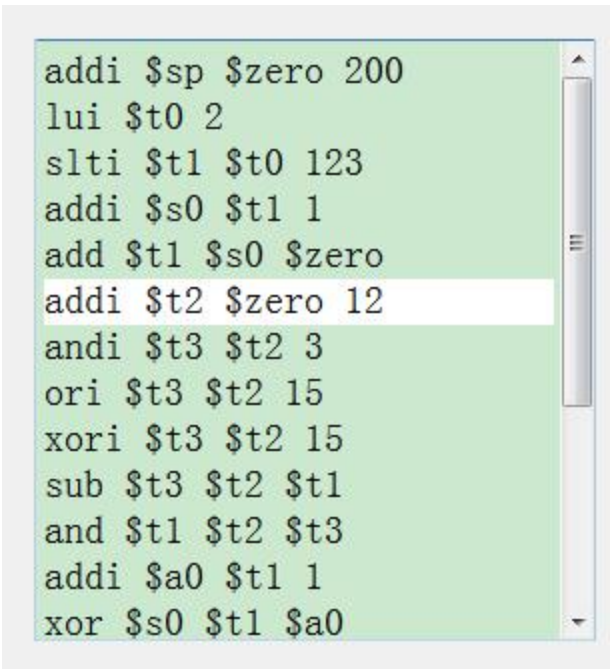
新建文件：

当你按下新建文件按钮，软件会自动将光标移动到文本框，准备接受输入



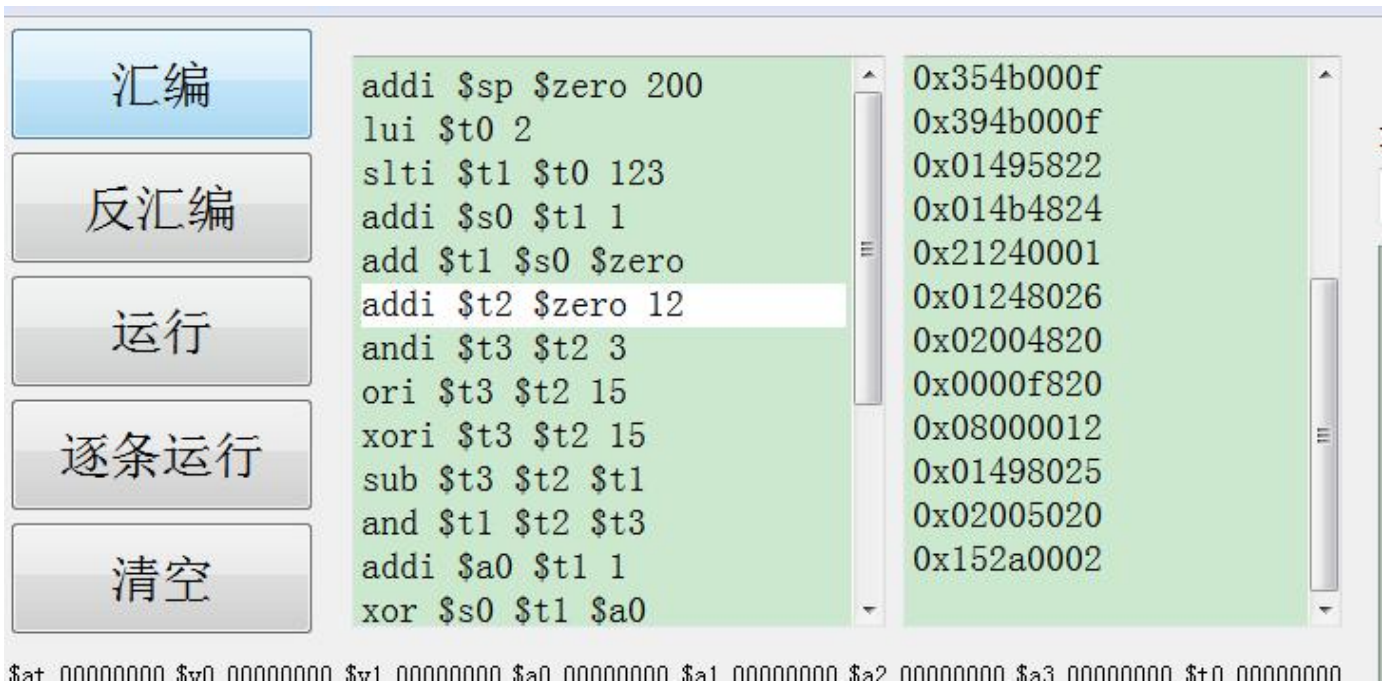
汇编：

您可以在文本框中输入您想要输入的 MIPS 指令



可以见到在您选中的那一行会高亮显示，方便查看

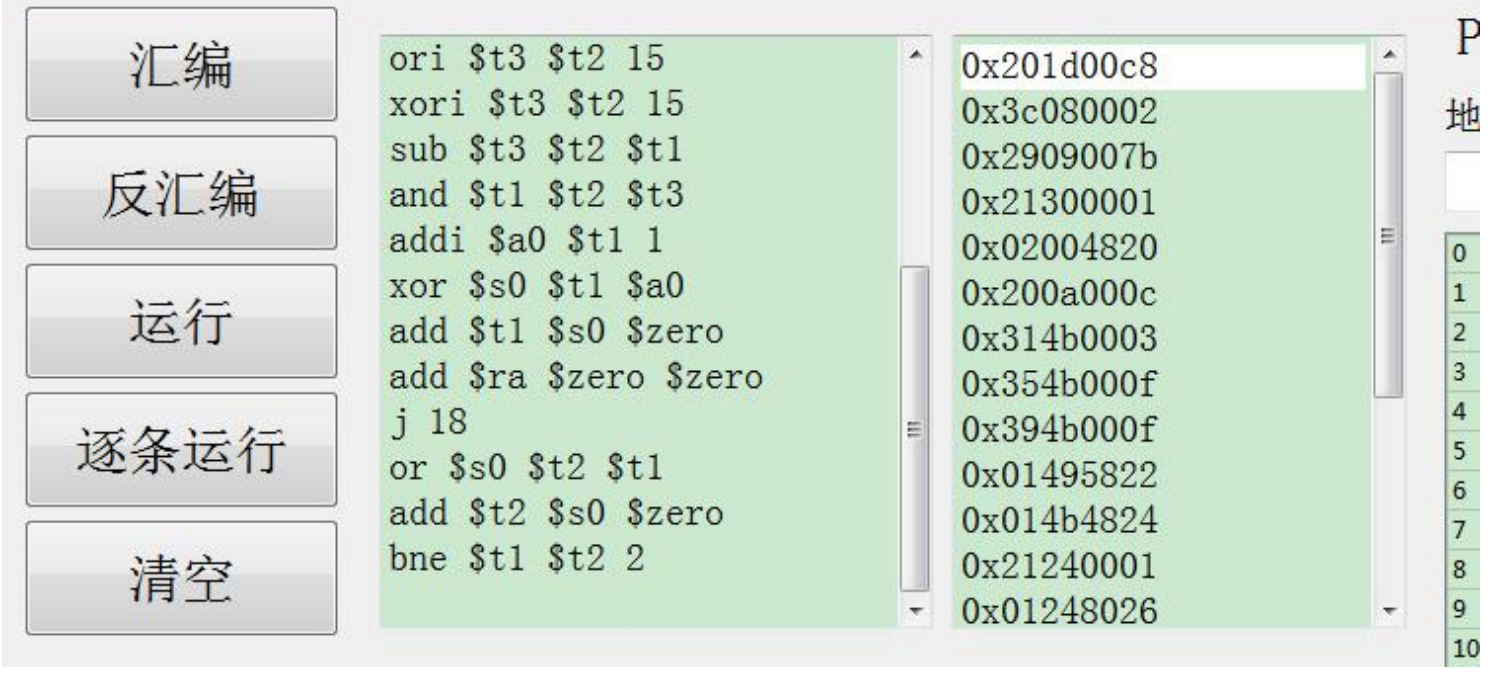
当你按下“汇编”按钮，在二进制文本框中会显示 16 进制的指令：



当您的指令输入错误时，软件会提醒您输入有误：

反汇编：

您可以将文本框中的 MIPS 指令清空 ,按下反汇编按钮 ,软件会把二进制指令反汇编成 MIPS 指令：



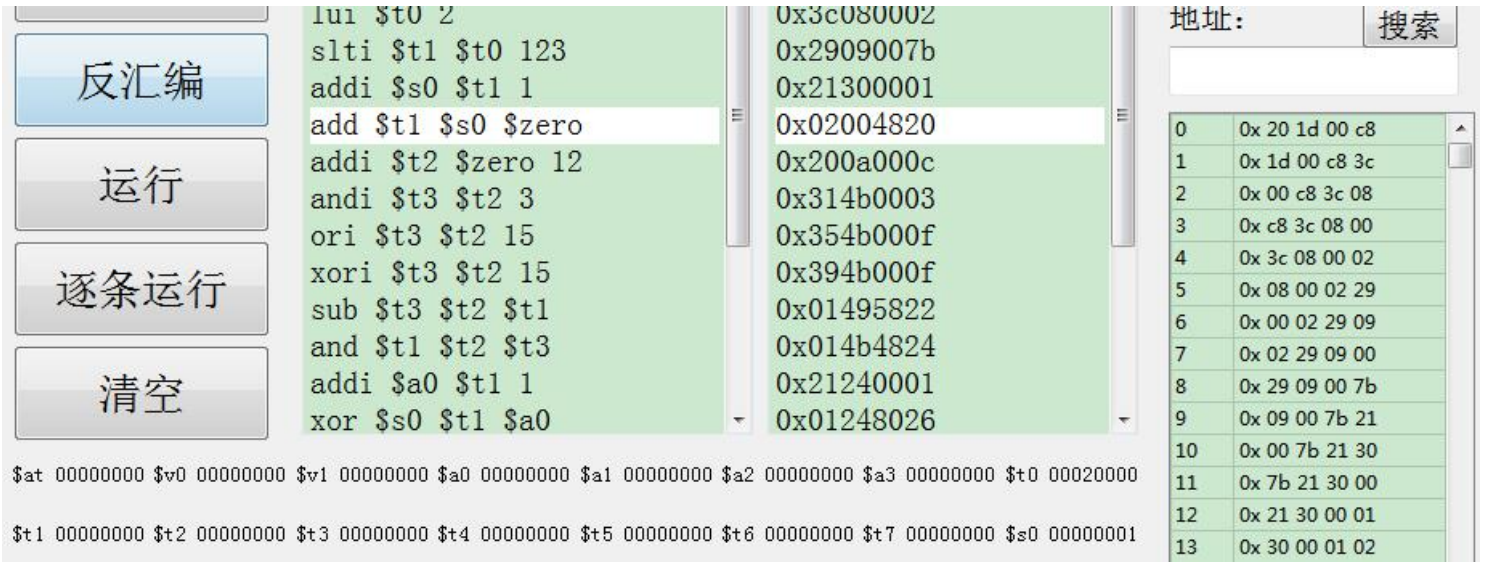
您可以试着检测正确性，一定会发现我们的软件永远不会出错

运行：

我们的软件可以模拟运行您的指令分为：逐条运行 和 一次性运行

逐条运行：

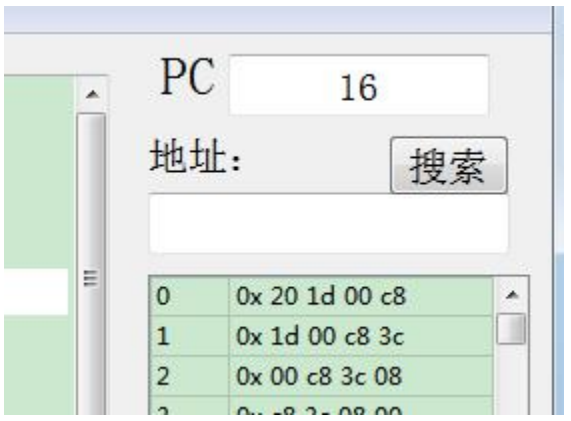
我们的软件支持逐条运行，您可以按下逐条运行的按钮来让软件运行下一条指令：



当您按下逐条运行的按钮，您会发现文本框的高亮行会提醒您当前运行的指令，您也会发现右侧的内存框的内容已经变化，

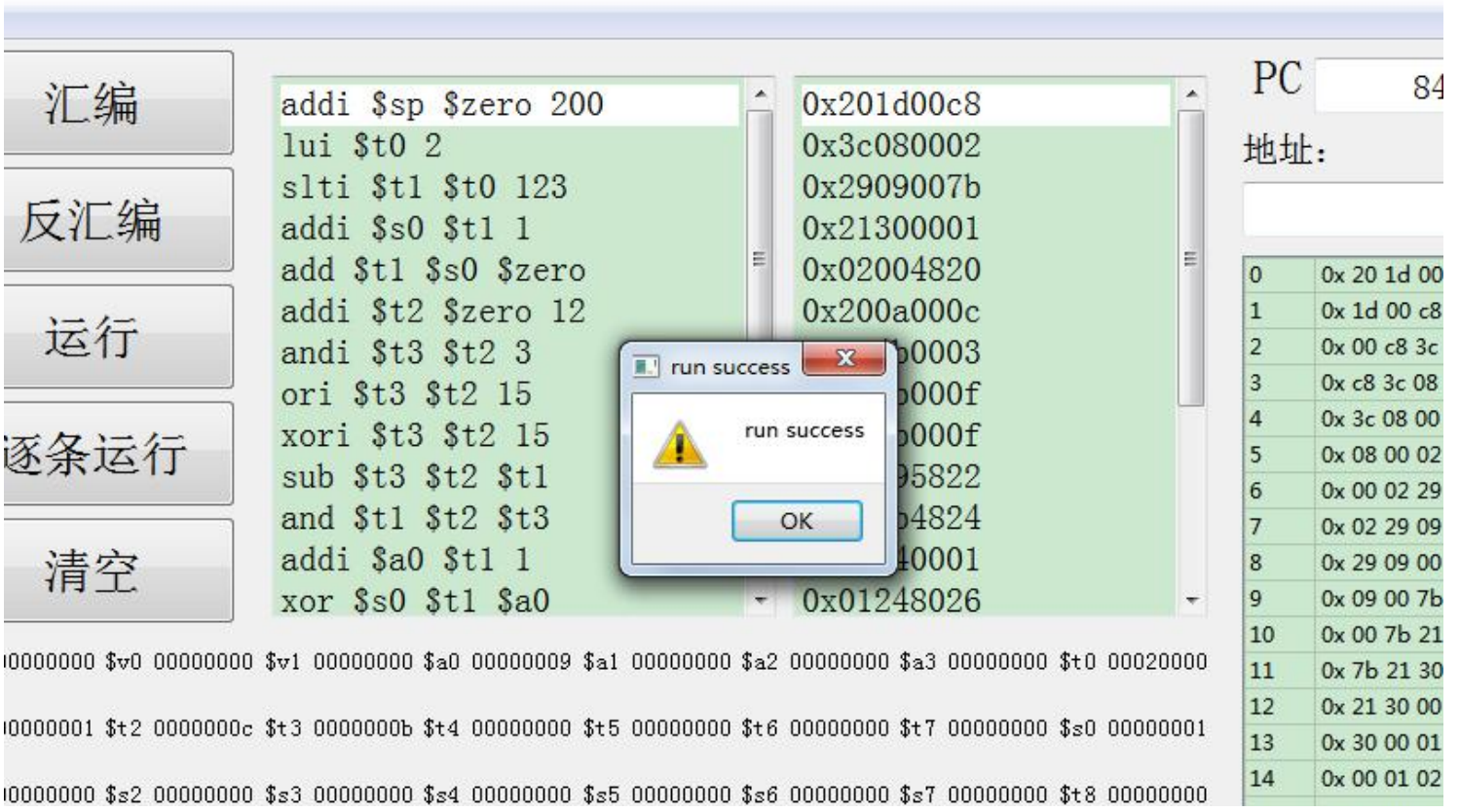
下面的 32 个寄存器的内容也根据您的指令而改变（如$s0,$t0）

PC 寄存器也相应的改变：



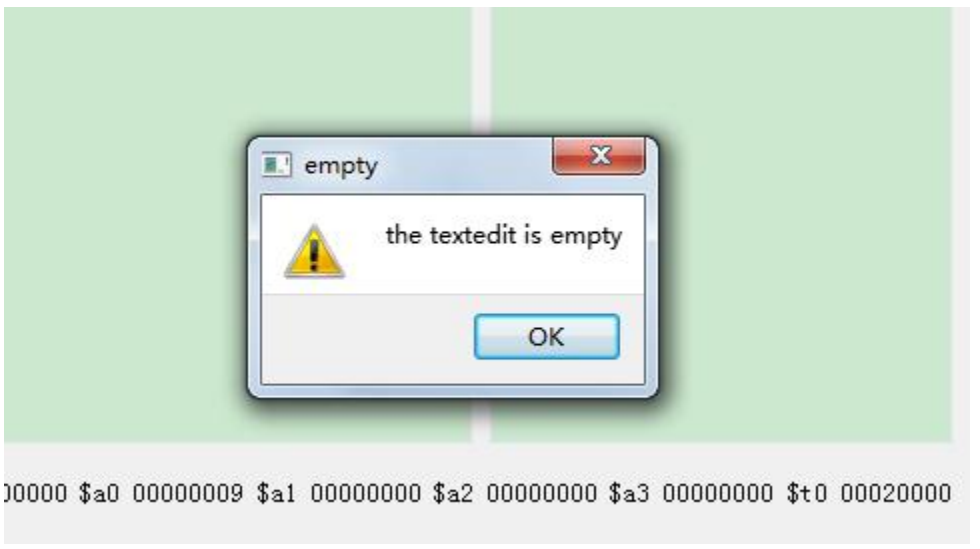
全部运行：

当您按下“运行”按钮，您的程序会一次性运行到底：



软件会提醒您程序运行成功

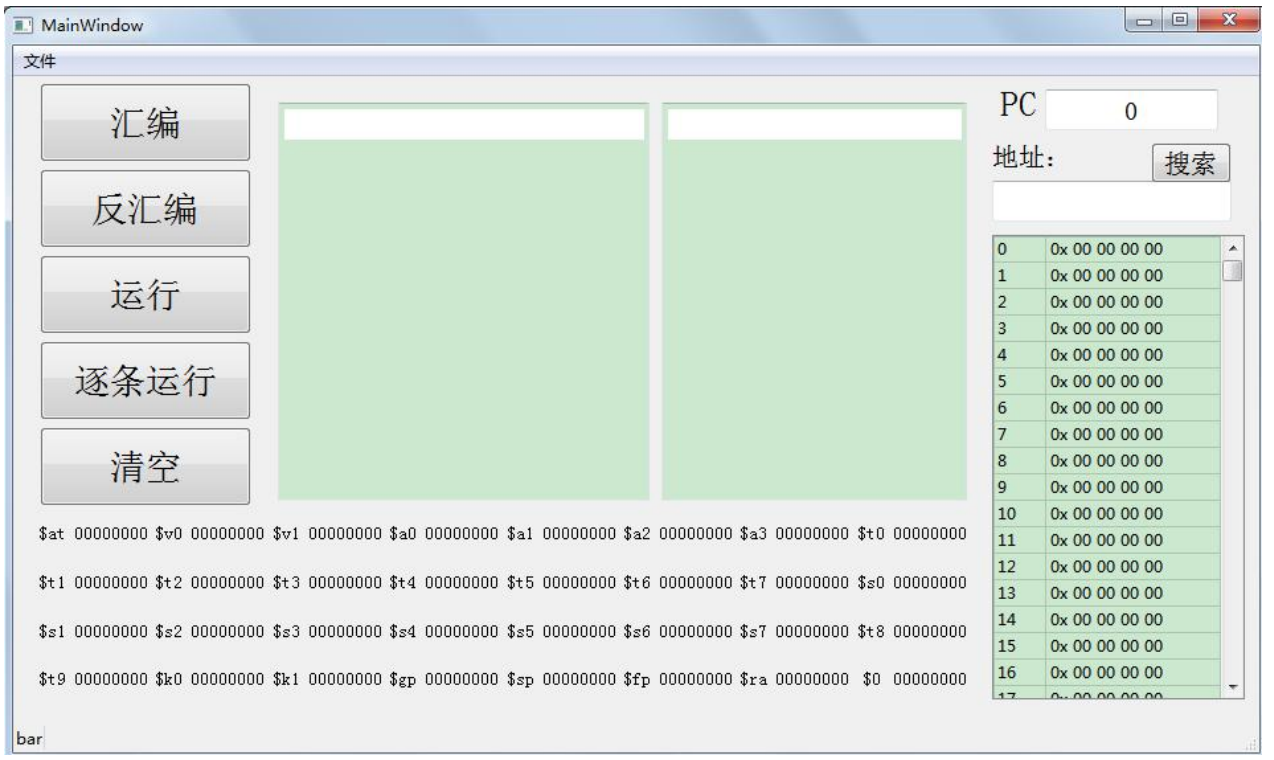
当您没有任何输入时，按下运行按钮，软件会提醒您文本框为空：



清空：

当你按下“清空”按钮：

则软件会初始化，内存，32 个寄存器，两个文本框都会清空

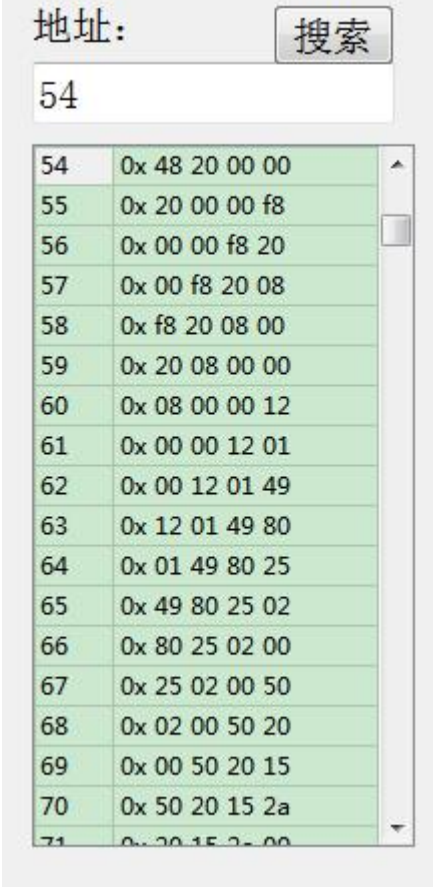


内存查看：

我们的软件暂时开辟了 1024*8 即 1K 的内存，以字节寻址

您可以在地址框中输入您想要查看的内存地址，按下“搜索”按钮

内存框会调到您想要的内存：



运行实例：

我们的软件支持 22 条指令，包括
add
sub
slt
and
or
nor
xor
addi
slti
andi
ori
xori
lw
sw
lb
sb
lui
beq
bne
j
jal
jr
我将以下面的 MIPS 代码作为测试实例：

```
addi $sp $zero 200
lui $t0 2
slti $t1 $t0 123
addi $s0 $t1 1
add $t1 $s0 $zero
addi $t2 $zero 12
andi $t3 $t2 3
ori $t3 $t2 15
xori $t3 $t2 15
```

sub $t3 $t2 $t1

and $t1 $t2 $t3

addi $a0 $t1 1

xor $s0 $t1 $a0

add $t1 $s0 $zero

add $ra $zero $zero

j 18

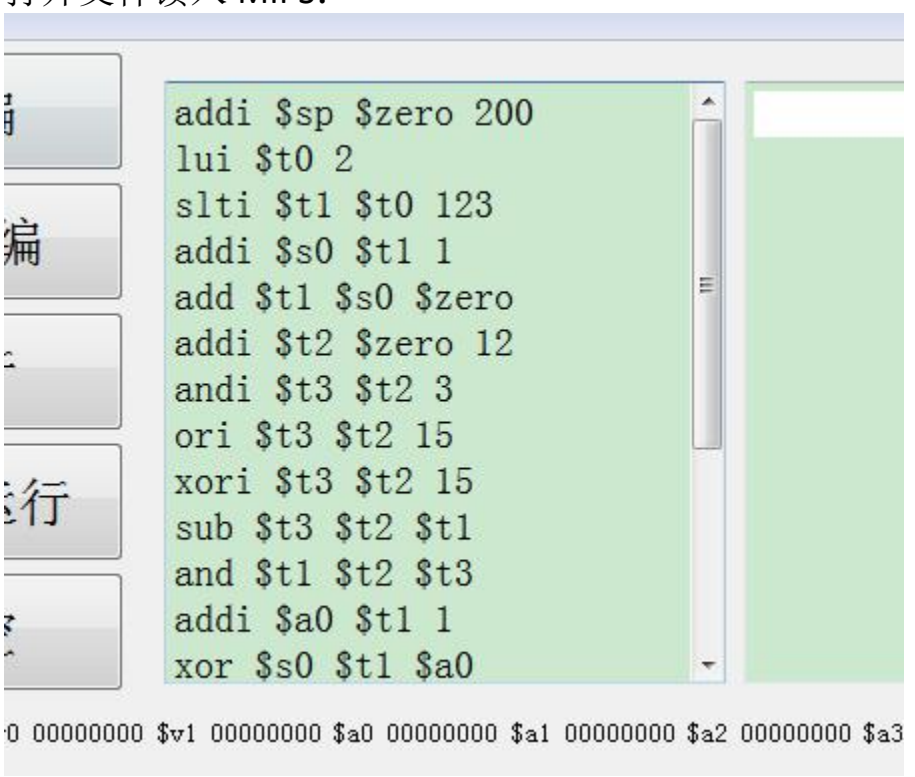
or $s0 $t2 $t1

add $t2 $s0 $zero

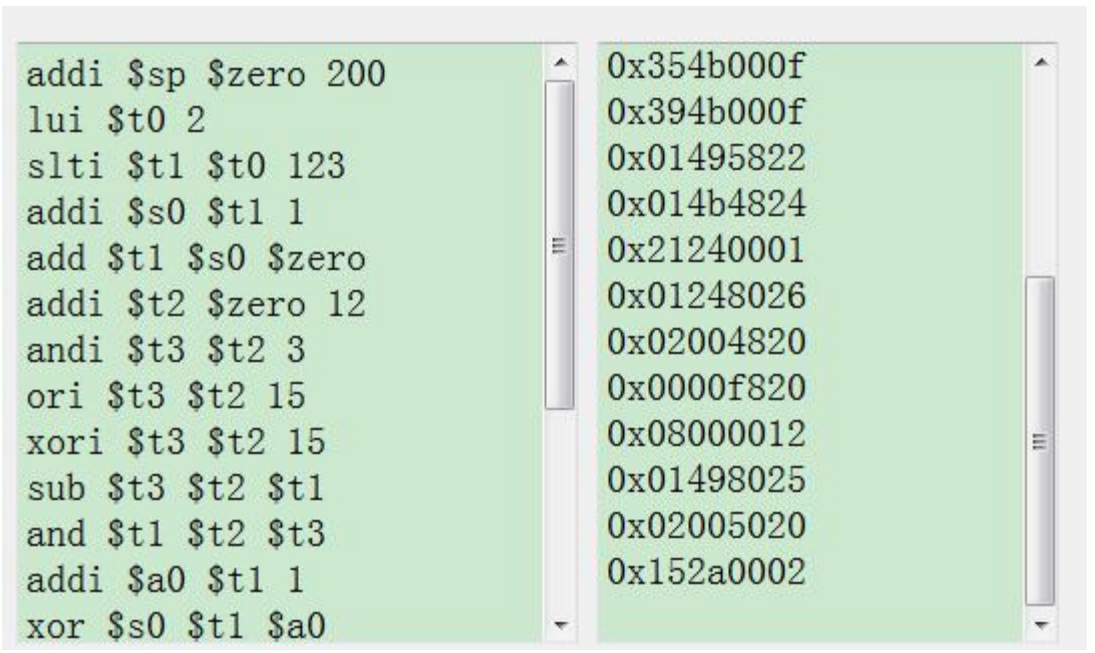
bne $t1 $t2 2

先清空所有：

打开文件读入 MIPS：

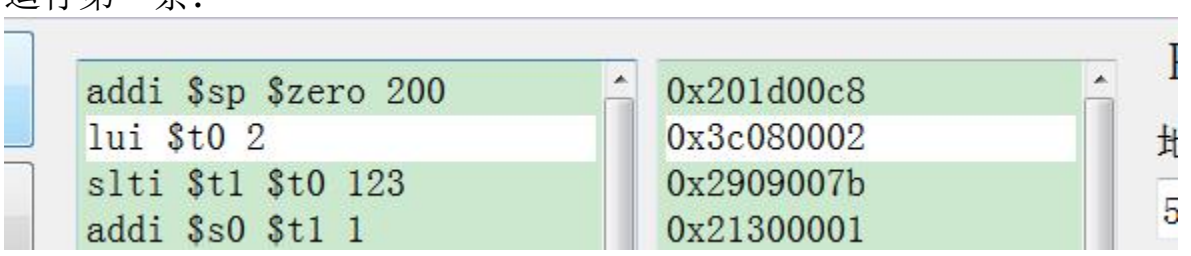


汇编之后：

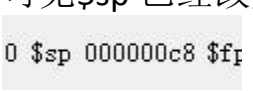


我们现在尝试着逐条运行：

运行第一条：

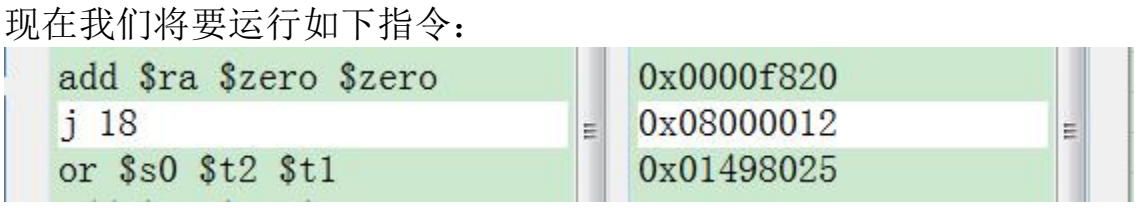


可见$sp 已经改变：



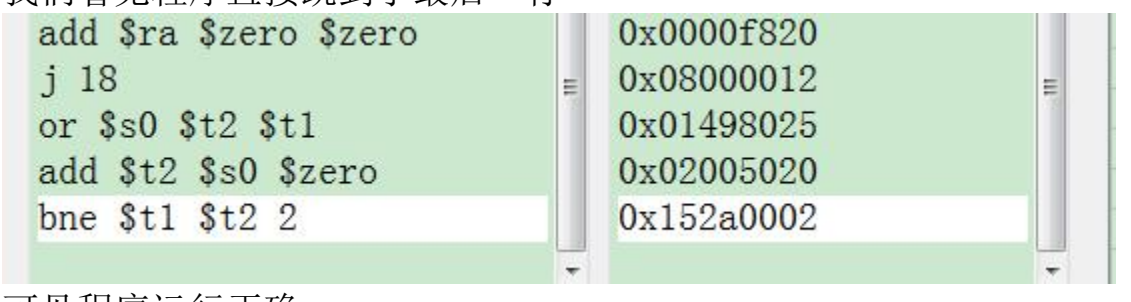
我们不断的逐条运行

现在我们将要运行如下指令：



按下逐条运行：

我们看见程序直接跳到了最后一行

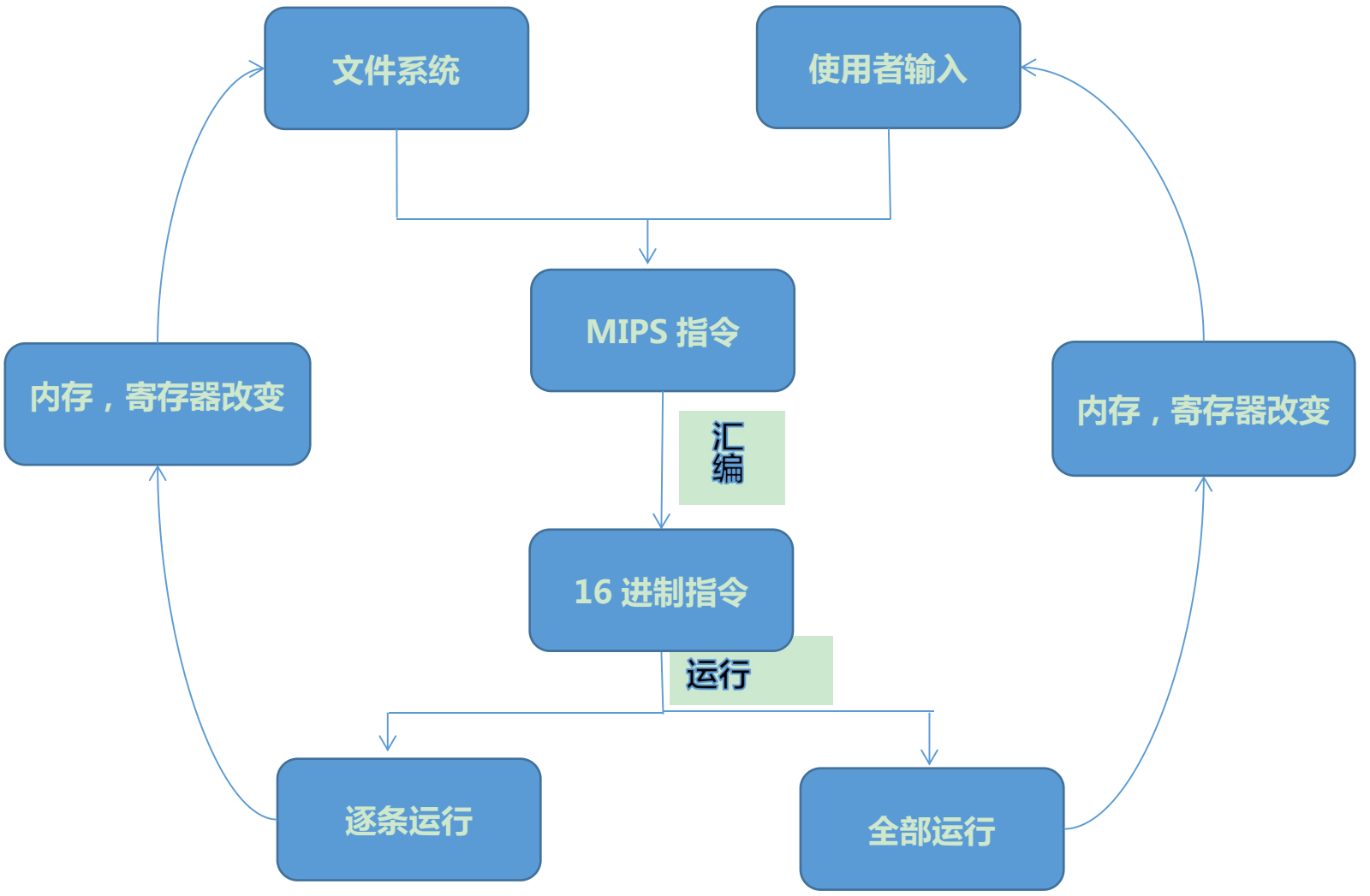


可见程序运行正确

此时我们的 32 个寄存器也相应的变化：

PC 也相应变化

软件流程图：



任务分工：

沈旭东：MIPS 指令的汇编与反汇编内核
王海荣：汇编与反汇编内核与界面的接口
黄一伦：MISP 运行内核
钱旭峰：内存寄存器类的实现，界面设计和实现，运行接口,内存管理

代码：

mai.h	汇编反汇编头文件
mainwindow.h	主界面头文件
mai.cpp	汇编反汇编实现文件
main.cpp	主函数文件
mainwindow.cpp	主界面实现文件
run.cpp	运行实现文件
mainwindow.ui	主界面界面文件

附录：

Mai.h：

```
#ifndef MAI_H
#define MAI_H

#endif // MAI_H
QString dodo(QString instring);
QString odod(QString instring);
```

Mainwindow.h：

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include<iostream>
#include <QString>
#include<string>
#include<QLabel>
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
    bool issaved;
    QString curfile;
    unsigned char memory[1024];
    unsigned int PC;
    unsigned int Register[32];
    QString IR;
    int INS;
    unsigned int lineindex;
    void update();
    QLabel *bar;
    bool flag;
    void showMessage(const QString& message);
    unsigned int add;
    QString changeR(unsigned int N);
```

```
public:
        QString change(int N);
        void gotoline( int line );

        void setIR();
            bool MainWindow_ins(unsigned int IR);
            void MainWindow_add(unsigned int IR);
            void MainWindow_sub(unsigned int IR);
            void MainWindow_slt(unsigned int IR);
            void MainWindow_and(unsigned int IR);
            void MainWindow_or(unsigned int IR);
            void MainWindow_nor(unsigned int IR);
            void MainWindow_xor(unsigned int IR);
            void MainWindow_addi(unsigned int IR);
            void MainWindow_slti(unsigned int IR);
            void MainWindow_andi(unsigned int IR);
            void MainWindow_ori(unsigned int IR);
            void MainWindow_xori(unsigned int IR);
            void MainWindow_lw(unsigned int IR);
            void MainWindow_sw(unsigned int IR);
            void MainWindow_lb(unsigned int IR);
            void MainWindow_sb(unsigned int IR);
            void MainWindow_lui(unsigned int IR);
            void MainWindow_beq(unsigned int IR);
            void MainWindow_bne(unsigned int IR);
            void MainWindow_j(unsigned int IR);
            void MainWindow_jal(unsigned int IR);
            void MainWindow_jr(unsigned int IR);
```

```
public slots:
    void clearall();
    void file_new(); //新建文件
    void file_saveornot(); //修改过的文件是否保存
    void file_save(); //保存文件
    void file_saveas(); //文件另存为
    bool savefile(const QString& fileName); //存储文件
    void file_open(); //打开文件
    bool file_load(const QString& fileName); //读取文件
    void sousuo();
    void on_assembler_clicked();
    void on_disassembler_clicked();
    void run();
    void runall();
    void highlightCurrentLine_1();
    void highlightCurrentLine_2();

};

#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"
```

```
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Mainwindow.cpp：

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "mai.h"
#include <QtGui>
#include <fstream>
#include<string>
#include<QMessageBox>
#include<QFileDialog>
#include<Qcolor>

MainWindow::MainWindow(QWidget *parent) :QMainWindow(parent),ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    issaved=0;
    curfile="未命名.mips";
    PC=0;
    IR="";
    INS=0;
    flag=0;
    add=0;
    lineindex=0;
    memset(memory, 0, 1024);
    memset(Register,0,32*4);

    bar=new QLabel("bar");
    statusBar()->addWidget(bar);
    bar->show();
    ui->pc->setFocusPolicy(Qt::NoFocus);
    ui->pc->setText(QString::number(PC));


    connect(ui->save,       SIGNAL(triggered(bool)),this,SLOT(file_save()));
    connect(ui->saveas,     SIGNAL(triggered(bool)),this,SLOT(file_saveas()));
    connect(ui->open,       SIGNAL(triggered(bool)),this,SLOT(file_open()));
    connect(ui->file_new,   SIGNAL(triggered(bool)),this,SLOT(file_new()));
    connect(ui->search,     SIGNAL(clicked(bool)),this,SLOT(sousuo()));
    connect(ui->run,        SIGNAL(clicked(bool)),this,SLOT(run()));
    connect(ui->runall,     SIGNAL(clicked(bool)),this,SLOT(runall()));
    connect(ui->text1,      SIGNAL(cursorPositionChanged()), this,
SLOT(highlightCurrentLine_1()));
    connect(ui->text2,      SIGNAL(cursorPositionChanged()), this,
SLOT(highlightCurrentLine_2()));
```

```
    connect(ui->wim,       SIGNAL(clicked(bool)),this,SLOT(clearall())  );
    update();



}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::gotoline( int line )
{
    QTextCursor tc = ui->text2->textCursor();
    int position = ui->text2->document()->findBlockByNumber ( line-1 ).position();
    tc.setPosition(position,QTextCursor::MoveAnchor);
    ui->text2->setTextCursor( tc );

    tc = ui->text1->textCursor();
    position = ui->text1->document()->findBlockByNumber ( line-1 ).position();
    tc.setPosition(position,QTextCursor::MoveAnchor);
    ui->text1->setTextCursor( tc );
}

void MainWindow::sousuo()
{
    int line=ui->addr->text().toInt();
    if(line<0 || line>1019)
    {
        QMessageBox::warning(this,"error",tr("请输入正确地址"));
        ui->addr->clear();
        return;
    }

    ui->MEM->setCurrentCell(line,0);
    return;
}

void MainWindow::clearall(){
   memset(this->memory, 0, 1024);
   memset(this->Register,0,32*4);
   this->PC=0;
   this->add=0;
   this->IR="";
   INS=0;
   add=0;
   flag=0;
   issaved=0;
   lineindex=0;
   ui->text1->setPlainText("");
   ui->text2->setPlainText("");
   statusBar()->showMessage("");
   update();
}

void MainWindow::showMessage(const QString& message)
```

```
{
    bar->setText(message);
}

void MainWindow::update()
{
    ui->pc->setText(QString::number(PC));
    ui->pc->repaint();
    ui->r0->setText(changeR(Register[0])); ui->r0->repaint();
    ui->r1->setText(changeR(Register[1])); ui->r1->repaint();
    ui->r2->setText(changeR(Register[2])); ui->r2->repaint();
    ui->r3->setText(changeR(Register[3])); ui->r3->repaint();
    ui->r4->setText(changeR(Register[4])); ui->r4->repaint();
    ui->r5->setText(changeR(Register[5])); ui->r5->repaint();
    ui->r6->setText(changeR(Register[6])); ui->r6->repaint();
    ui->r7->setText(changeR(Register[7])); ui->r7->repaint();
    ui->r8->setText(changeR(Register[8])); ui->r8->repaint();
    ui->r9->setText(changeR(Register[9])); ui->r9->repaint();
    ui->r10->setText(changeR(Register[10])); ui->r10->repaint();
    ui->r11->setText(changeR(Register[11])); ui->r11->repaint();
    ui->r12->setText(changeR(Register[12])); ui->r12->repaint();
    ui->r13->setText(changeR(Register[13])); ui->r13->repaint();
    ui->r14->setText(changeR(Register[14])); ui->r14->repaint();
    ui->r15->setText(changeR(Register[15])); ui->r15->repaint();
    ui->r16->setText(changeR(Register[16])); ui->r16->repaint();
    ui->r17->setText(changeR(Register[17])); ui->r17->repaint();
    ui->r18->setText(changeR(Register[18])); ui->r18->repaint();
    ui->r19->setText(changeR(Register[19])); ui->r19->repaint();
    ui->r20->setText(changeR(Register[20])); ui->r20->repaint();
    ui->r21->setText(changeR(Register[21])); ui->r21->repaint();
    ui->r22->setText(changeR(Register[22])); ui->r22->repaint();
    ui->r23->setText(changeR(Register[23])); ui->r23->repaint();
    ui->r24->setText(changeR(Register[24])); ui->r24->repaint();
    ui->r25->setText(changeR(Register[25])); ui->r25->repaint();
    ui->r26->setText(changeR(Register[26])); ui->r26->repaint();
    ui->r27->setText(changeR(Register[27])); ui->r27->repaint();
    ui->r28->setText(changeR(Register[28])); ui->r28->repaint();
    ui->r29->setText(changeR(Register[29])); ui->r29->repaint();
    ui->r30->setText(changeR(Register[30])); ui->r30->repaint();
    ui->r31->setText(changeR(Register[31])); ui->r31->repaint();


    ui->MEM->clear();
    ui->MEM->setColumnCount(2);
    ui->MEM->setRowCount(1020);
    ui->MEM->setColumnWidth(0,40);
    ui->MEM->setColumnWidth(1,100);
    ui->MEM->verticalHeader()->setVisible(false);   //隐藏列表头
    ui->MEM->horizontalHeader()->setVisible(false); //隐藏列表头
    ui->MEM->horizontalHeader()->setStretchLastSection(true); //设置充满表宽度
    ui->MEM->setEditTriggers(QAbstractItemView::NoEditTriggers);
    int cou;
    int num;
    for(cou = 0; cou < 1020; cou ++)
    {
        ui->MEM->setRowHeight(cou,20);
        ui->MEM->setItem(cou, 0, new QTableWidgetItem(QString::number(cou)));
```

```
        num=memory[cou]<<24 | memory[cou+1]<<16 |memory[cou+2]<<8 | memory[cou+3];
        ui->MEM->setItem(cou, 1, new QTableWidgetItem(change(num)));
    }


}

QString MainWindow::changeR(unsigned int N)
{
    QString res=QString::number(N,16);
    while(res.length()<8)
        res.insert(0,"0");

    while(res.length()>8)
        res.remove(0,1);
    return res;
}

QString MainWindow::change(int N)
{
    QString res=QString::number(N,16);
    while(res.length()<8)
        res.insert(0,"0");

    while(res.length()>8)
        res.remove(0,1);

    res.insert(2," ");
    res.insert(5," ");
    res.insert(8," ");
    res.insert(0,"0x ");
    return res;
}

void MainWindow::file_save()
{
    if(issaved)
        savefile(curfile);
    else
        file_saveas();
}

bool MainWindow::savefile(const QString& filename)
{
    QFile file(filename);
    if(!file.open(QFile::WriteOnly | QFile::Text))
    {
        QMessageBox::warning(this,"save file",tr("cannot
save %1:\n %2").arg(filename).arg(file.errorString()));
        return false;
    }

    QTextStream out(&file);
    out<<ui->text1->toPlainText();
    issaved=1;
    curfile=QFileInfo(filename).canonicalFilePath();
    setWindowTitle(curfile);
```

```
    return true;


}

void MainWindow::file_saveas()
{
    QString filename=QFileDialog::getSaveFileName(this,tr("save as"),curfile);
    //获得文件名

    if(!filename.isEmpty())
    {
        savefile(filename);
    }

}

void MainWindow::file_saveornot()
{
    if(ui->text1->document()->isModified())
    {
        QMessageBox box;
        box.setWindowTitle("save or not ?");
        box.setIcon(QMessageBox::Warning);
        box.setText(curfile + " has not saved,save now?");
        box.setStandardButtons(QMessageBox::Yes | QMessageBox::No);
        if(box.exec()==QMessageBox::Yes)
            file_save();
    }
}

void MainWindow::file_new()
{
    file_saveornot();
    issaved=false;
    curfile="未命名.mips";
    ui->text1->clear();
    ui->text2->clear();
    ui->text1->setVisible(true);
    ui->text2->setVisible(true);
}

void MainWindow::file_open()
{
    file_saveornot();
    QString filename=QFileDialog::getOpenFileName(this);
    if(!filename.isEmpty())//如果文件名不为空
        file_load(filename);

    issaved=true;
    ui->text1->setVisible(true);

}

bool MainWindow::file_load(const QString& filename)
{
```

```
    QFile file(filename);
    if(!file.open(QFile::ReadOnly | QFile::Text))
    {
        QMessageBox::warning(this,"open file",tr("cannot
open %1:\n %2").arg(filename).arg(file.errorString()));
        return false;
    }

    QTextStream in(&file);
    ui->text1->setPlainText(in.readAll());
    curfile = QFileInfo(filename).canonicalFilePath();
    setWindowTitle(curfile);
    return true;
}

void MainWindow::on_assembler_clicked()
{
    QString q;

    int linenum=ui->text1->document()->lineCount();
    for(int i=0;i<linenum;i++)
    {
        QString str = ui->text1->document()->findBlockByLineNumber(i).text();
        if(!str.isEmpty())
        {
        q =dodo(str);

        if(!i)
            ui->text2->setPlainText(q);
        else
            ui->text2->appendPlainText(q);

        }



    }
    PC=0;
}

void MainWindow::on_disassembler_clicked()
{
    int linenum=ui->text2->document()->lineCount();
    QString q,p;
    unsigned long ins;
    bool ok;
    for(int i=0;i<linenum;i++)
    {
        QString str = ui->text2->document()->findBlockByLineNumber(i).text();
        q=str;
        q.remove("0x");
        p=q.mid(4,4);
        q=q.mid(0,4);

        ins=p.toInt(&ok,16);
        memory[4*i+2]=ins>>8;
        memory[4*i+3]=ins;
```

```
        ins=q.toInt(&ok,16);
        memory[4*i]=ins>>8;
        memory[4*i+1]=ins;
        if(!str.isEmpty())
        {
        QString q =odod(str);

        if(!i)
            ui->text1->setPlainText(q);
        else
            ui->text1->appendPlainText(q);
        }
    }
}

void MainWindow::run()
{
    if(ui->text2->toPlainText()=="" && ui->text1->toPlainText()!="")
    {
              on_assembler_clicked();
              on_disassembler_clicked();
              flag=1;
    }
    else if(ui->text1->toPlainText()=="")
    {
        QMessageBox::warning(this,"empty",tr("the textedit is empty"));
        return;
    }
    else if(flag==0)
    {
        on_disassembler_clicked();
        flag=1;
    }

    unsigned int linenum=ui->text2->document()->lineCount();

    if(lineindex==linenum)
    {
        lineindex=0;
        flag=0;
        statusBar()->showMessage("正在运行行号:"+QString::number(lineindex));
        QMessageBox::warning(this,"end of running",tr("end of running"));
        return;
    }

    INS=memory[PC]<<24 | memory[PC+1]<<16 |memory[PC+2]<<8 | memory[PC+3];

    if(!MainWindow::MainWindow_ins(INS))
    {
        QMessageBox::warning(this,"run error",tr("cannot run %1").arg(IR));
        return;
    }
    statusBar()->showMessage("正在运行行号:"+QString::number(lineindex+2));

    lineindex=PC/4;
    gotoline(lineindex+1);
    update();
```

```
    return;
}

void MainWindow::runall()
{
    PC=0;
    if(ui->text2->toPlainText()=="" && ui->text1->toPlainText()!="")
    {
              on_assembler_clicked();
              on_disassembler_clicked();
              flag=1;
    }
    else if(ui->text1->toPlainText()=="")
    {
        QMessageBox::warning(this,"empty",tr("the textedit is empty"));
        return;
    }
    else if(flag==0)
    {
        on_disassembler_clicked();
        flag=1;
    }
    unsigned int linenum=ui->text2->document()->lineCount();
    lineindex=0;


    while(lineindex<linenum)
    {
        INS=memory[PC]<<24 | memory[PC+1]<<16 |memory[PC+2]<<8 | memory[PC+3];
        if(!MainWindow::MainWindow_ins(INS))
        {
            update();
            QMessageBox::warning(this,"run error",tr("cannot run %1").arg(IR));
            return;
        }
        statusBar()->showMessage("正在运行行号:"+QString::number(lineindex+2));
        gotoline(lineindex+2);


        lineindex=PC/4;
    }

    update();
    QMessageBox::warning(this,"run success",tr("run success"));
    flag=0;
    return;


}


void MainWindow::highlightCurrentLine_1()
  {
     QList<QTextEdit::ExtraSelection> extraSelections;

     if (!ui->text1->isReadOnly()) {
```

```
        QTextEdit::ExtraSelection selection;

        QColor lineColor = QColor(Qt::blue).lighter(200);

        selection.format.setBackground(lineColor);
        selection.format.setProperty(QTextFormat::FullWidthSelection, true);
        selection.cursor = ui->text1->textCursor();
        selection.cursor.clearSelection();
        extraSelections.append(selection);
    }

    ui->text1->setExtraSelections(extraSelections);
 }

void MainWindow::highlightCurrentLine_2()
 {
    QList<QTextEdit::ExtraSelection> extraSelections;

    if (!ui->text2->isReadOnly()) {
        QTextEdit::ExtraSelection selection;

        QColor lineColor = QColor(Qt::blue).lighter(200);

        selection.format.setBackground(lineColor);
        selection.format.setProperty(QTextFormat::FullWidthSelection, true);
        selection.cursor = ui->text2->textCursor();
        selection.cursor.clearSelection();
        extraSelections.append(selection);
    }

    ui->text2->setExtraSelections(extraSelections);
 }
```

Run.cpp:

```
#include "mainwindow.h"

void MainWindow::setIR(){
    this->IR=this->memory[this->add+this->PC];
}

bool MainWindow::MainWindow_ins(unsigned int IR){
    switch(IR>>26){
     case 0x00:{
        switch(IR&0x3f){
         case 0x20:MainWindow::MainWindow_add(IR);break;
         case 0x22:MainWindow::MainWindow_sub(IR);break;
         case 0x24:MainWindow::MainWindow_and(IR);break;
         case 0x25:MainWindow::MainWindow_or(IR);break;
         case 0x27:MainWindow::MainWindow_nor(IR);break;
         case 0x26:MainWindow::MainWindow_xor(IR);break;
         case 0x2a:MainWindow::MainWindow_slt(IR);break;
         case 0x08:MainWindow::MainWindow_jr(IR);break;
         default:return false;//错误
        }break;
```

```
    }
    case 0x08:MainWindow::MainWindow_addi(IR);break;
    case 0x0a:MainWindow::MainWindow_slti(IR);break;
    case 0x0c:MainWindow::MainWindow_andi(IR);break;
    case 0x0d:MainWindow::MainWindow_ori(IR);break;
    case 0x0e:MainWindow::MainWindow_xori(IR);break;
    case 0x23:MainWindow::MainWindow_lw(IR);break;
    case 0x2b:MainWindow::MainWindow_sw(IR);break;
    case 0x20:MainWindow::MainWindow_lb(IR);break;
    case 0x28:MainWindow::MainWindow_sb(IR);break;
    case 0x0f:if(IR>>21&0x1f)return false;else MainWindow::MainWindow_lui(IR);break;
    case 0x04:MainWindow::MainWindow_beq(IR);break;
    case 0x05:MainWindow::MainWindow_bne(IR);break;
    case 0x02:MainWindow::MainWindow_j(IR);break;
    case 0x03:MainWindow::MainWindow_jal(IR);break;
    default:return false;//错误
   }
   return true;
}


void MainWindow::MainWindow_add(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]+this->Register[(IR>>16)&0x1f];
   this->PC+=4;

}

void MainWindow::MainWindow_sub(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]-this->Register[(IR>>16)&0x1f];
    this->PC+=4;
}
void MainWindow::MainWindow_slt(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]<this->Register[(IR>>16)&0x1f]?1:
0;
   this->PC+=4;
}

void MainWindow::MainWindow_and(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]&this->Register[(IR>>16)&0x1f];
   this->PC+=4;
}
void MainWindow::MainWindow_or(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]|this->Register[(IR>>16)&0x1f];
   this->PC+=4;
}

void MainWindow::MainWindow_nor(unsigned int IR){

this->Register[(IR>>11)&0x1f]=~(this->Register[(IR>>21)&0x1f]|this->Register[(IR>>16)&0x1f])
;
   this->PC+=4;
}
```

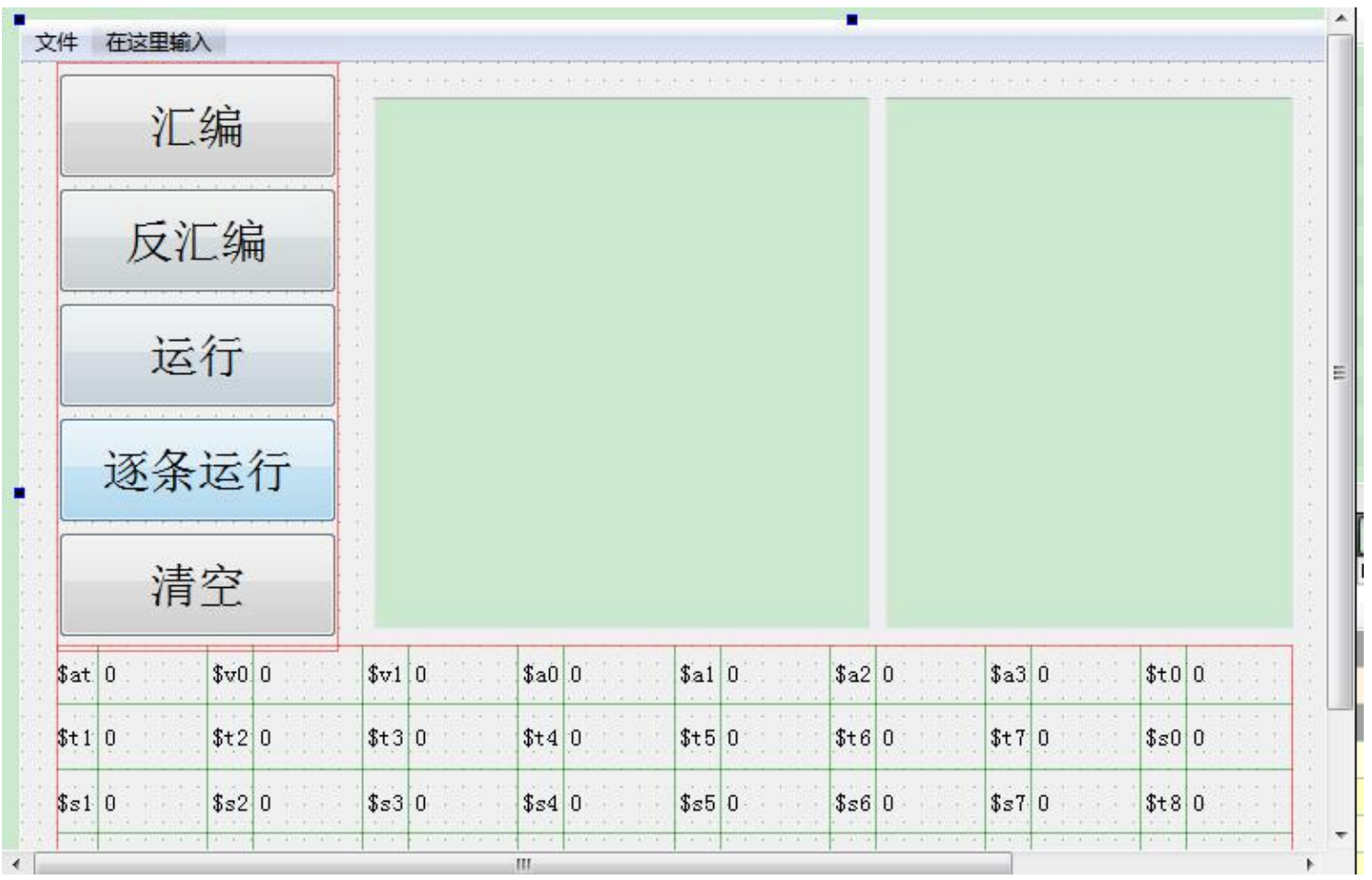
```
void MainWindow::MainWindow_xor(unsigned int IR){

this->Register[(IR>>11)&0x1f]=this->Register[(IR>>21)&0x1f]^this->Register[(IR>>16)&0x1f];
    this->PC+=4;
}

void MainWindow::MainWindow_addi(unsigned int IR){
    unsigned int tmp,sign;
    tmp=IR&0xffff;
    sign=((tmp>>15)?0xffff:0x0000)<<16;
    this->Register[(IR>>16)&0x1f]=this->Register[(IR>>21)&0x1f]+tmp+sign;
    this->PC+=4;
}

void MainWindow::MainWindow_slti(unsigned int IR){
    unsigned int tmp,sign;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    this->Register[(IR>>16)&0x1f]=this->Register[(IR>>21)&0x1f]<(tmp+sign)?1:0;
    this->PC+=4;
}

void MainWindow::MainWindow_andi(unsigned int IR){
    unsigned int tmp,sign;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    this->Register[(IR>>16)&0x1f]=this->Register[(IR>>21)&0x1f]&(tmp+sign);
    this->PC+=4;
}
```

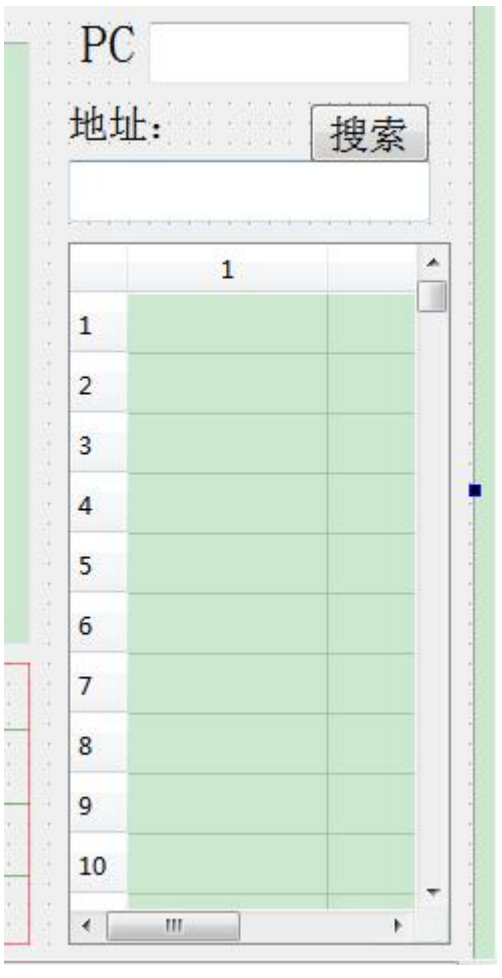
```
void MainWindow::MainWindow_ori(unsigned int IR){
    unsigned int tmp,sign;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    this->Register[(IR>>16)&0x1f]=this->Register[(IR>>21)&0x1f]|(tmp+sign);
    this->PC+=4;
}

void MainWindow::MainWindow_xori(unsigned int IR){
    unsigned int tmp,sign;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    this->Register[(IR>>16)&0x1f]=this->Register[(IR>>21)&0x1f]^(tmp+sign);
    this->PC+=4;
}

void MainWindow::MainWindow_lw(unsigned int IR){
    unsigned int tmp,sign,res,address;
    res=0;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    address=tmp+sign+this->Register[(IR>>21)&0x1f];
    res+=(this->memory[address])&0xff;
    res=((res<<8)+this->memory[address+1])&0xffff;
    res=((res<<8)+this->memory[address+2])&0xffffff;
    res=((res<<8)+this->memory[address+3])&0xffffffff;
    this->Register[(IR>>16)&0x1f]=res;
```

```
    this->PC+=4;
}
void MainWindow::MainWindow_sw(unsigned int IR){
    unsigned int tmp,sign,address;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    address=tmp+sign+this->Register[(IR>>21)&0x1f];
    this->memory[address]=Register[(IR>>16)&0x1f]>>24;
    this->memory[address+1]=(Register[(IR>>16)&0x1f]>>16)&0xff;
    this->memory[address+2]=(Register[(IR>>16)&0x1f]>>8)&0xff;
    this->memory[address+3]=(Register[(IR>>16)&0x1f])&0xff;
    this->PC+=4;
}

void MainWindow::MainWindow_lb(unsigned int IR){
    unsigned int tmp,sign,res,address;
    res=0;
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    address=tmp+sign+this->Register[(IR>>21)&0x1f];
    res+=this->memory[address+3];
    sign=((res>>7)?0xffffff:0x000000)<<8;
    this->Register[(IR>>16)&0x1f]=res+sign;
    this->PC+=4;
}

void MainWindow::MainWindow_sb(unsigned int IR){
    unsigned int tmp,res,sign,address;
    res=Register[(IR>>16)&0xff];
    tmp=IR&0xffff;
    sign=(tmp>>15?0xffff:0x0000)<<16;
    address=tmp+sign+this->Register[(IR>>21)&0x1f];
    this->memory[address]=(res>>7)?0xff:0x00;
    this->memory[address+1]=(res>>7)?0xff:0x00;
    this->memory[address+2]=(res>>7)?0xff:0x00;
    this->memory[address+3]=res;
    this->PC+=4;
}

void MainWindow::MainWindow_lui(unsigned int IR){
   this->Register[(IR>>16)&0x1f]=(IR&0xffff)<<16;
    this->PC+=4;
}
void MainWindow::MainWindow_beq(unsigned int IR){
    unsigned int tmp,sign,offset;
    tmp=IR&0xffff;
    sign=((tmp>>15)?0x3fff:0x0000)<<16;
    offset=(tmp+sign)<<2;

this->PC=(this->Register[(IR>>16)&0x1f]==this->Register[(IR>>21)&0x1f])?this->PC+4+offset:th
is->PC+4;
}

void MainWindow::MainWindow_bne(unsigned int IR){
    unsigned int tmp,sign,offset;
    tmp=IR&0xffff;
    sign=((tmp>>15)?0x3fff:0x0000)<<16;
```

```
    offset=(tmp+sign)<<2;

this->PC=(this->Register[(IR>>16)&0x1f]!=this->Register[(IR>>21)&0x1f])?this->PC+4+offset:th
is->PC+4;
}

void MainWindow::MainWindow_j(unsigned int IR){
    unsigned int address;
    address=((this->PC)&0xf0000000)+(IR&0x3ffff)<<2;
    this->PC=address;
}
void MainWindow::MainWindow_jal(unsigned int IR){
    unsigned int address;
    address=((this->PC)&0xf0000000)+(IR&0x3ffff)<<2;
    this->memory[31]=this->PC+4;
    this->PC=address;
}
void MainWindow::MainWindow_jr(unsigned int IR){
    this->PC=this->Register[(IR>>21)&0x1f];
}
```

mainwindow.ui





Mai.cpp:

```
#include<iostream>
#include<string>
#include<cstring>
#include<QString>
#include"mai.h"

using namespace std;
typedef unsigned int bit32;

class instruction{
private:
    bit32 binary_code;
    char* str_code;

    int str2bin_add(char* in){
        int i = 0, r, p = 0;
        bit32 binary = 0;
        char *temp;
        temp = (char*)calloc(7, 1);
        while (in[p] != ' '){
            temp[i] = in[p];
            i++;
            p++;
            if (i > 6)break;
        }
        temp[i] = 0;
        if ((r = read_register(temp)) == -1)return -1;
        binary += bit32(r) << 11;
        p++;
```

```
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    binary += 0x20;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_sub(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 11;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
```

```
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    binary += 0x22;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_slt(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 11;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    binary += 0x2a;
    this->binary_code = binary;
    binary_code2str_code();
```

```
    return 0;
}

int str2bin_and(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 11;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    binary += 0x24;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_or(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
```

```
}
temp[i] = 0;
if ((r = read_register(temp)) == -1)return -1;
binary += bit32(r) << 11;
p++;
i = 0;
while (in[p] != ' '){
    temp[i] = in[p];
    i++;
    p++;
    if (i > 6)break;
}
temp[i] = 0;
if ((r = read_register(temp)) == -1)return -1;
binary += bit32(r) << 21;
p++;
i = 0;
while (in[p] != 0){
    temp[i] = in[p];
    i++;
    p++;
    if (i > 6)break;
}
temp[i] = 0;
if ((r = read_register(temp)) == -1)return -1;
binary += bit32(r) << 16;
binary += 0x25;
this->binary_code = binary;
binary code2str code();
```

```
    return 0;
}

int str2bin_nor(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 11;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
```

```
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    binary += 0x27;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_xor(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 11;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != 0){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
```

```
    binary += 0x26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_addi(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    sscanf(in + p, "%d%c", &immediate);
    binary += immediate & 0xffff;
    binary += (0x8) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_slti(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
```

```
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0xa) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_andi(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0xc) << 26;
    this->binary_code = binary;
    binary_code2str_code();
```

```
    return 0;
}

int str2bin_ori(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0xd) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_xori(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
```

```
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0xe) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_lw(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp, *temp1;
    temp = (char*)calloc(7, 1);
    temp1 = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d(%s", &immediate,temp1);
    binary += immediate & 0xffff;
    i = 0;
    while (temp1[i] != ')'){
        temp[i] = temp1[i];
        i++;
        if (i > 6)break;
    }
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    binary += (0x23) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}
```

```
int str2bin_sw(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp, *temp1;
    temp = (char*)calloc(7, 1);
    temp1 = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d(%s", &immediate, temp1);
    binary += immediate & 0xffff;
    i = 0;
    while (temp1[i] != ')'){
        temp[i] = temp1[i];
        i++;
        if (i > 6)break;
    }
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    binary += (0x2b) << 26;
    this->binary code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_lb(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp, *temp1;
    temp = (char*)calloc(7, 1);
    temp1 = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d(%s", &immediate, temp1);
    binary += immediate & 0xffff;
    i = 0;
    while (temp1[i] != ')'){
        temp[i] = temp1[i];
```

```
        i++;
        if (i > 6)break;
    }
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    binary += (0x20) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_sb(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp, *temp1;
    temp = (char*)calloc(7, 1);
    temp1 = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d(%s", &immediate, temp1);
    binary += immediate & 0xffff;
    i = 0;
    while (temp1[i] != ')'){
        temp[i] = temp1[i];
        i++;
        if (i > 6)break;
    }
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    binary += (0x28) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_lui(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
```

```
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0xf) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_beq(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    p++;
    i = 0;
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 16;
    p++;
    sscanf(in + p, "%d", &immediate);
    binary += immediate & 0xffff;
    binary += (0x4) << 26;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

int str2bin_bne(char* in){
    int i = 0, r, p = 0;
    bit32 binary = 0;
```

```
int immediate = 0;
char *temp;
temp = (char*)calloc(7, 1);
while (in[p] != ' '){
    temp[i] = in[p];
    i++;
    p++;
    if (i > 6)break;
}
temp[i] = 0;
if ((r = read_register(temp)) == -1)return -1;
binary += bit32(r) << 21;
p++;
i = 0;
while (in[p] != ' '){
    temp[i] = in[p];
    i++;
    p++;
    if (i > 6)break;
}
temp[i] = 0;
if ((r = read_register(temp)) == -1)return -1;
binary += bit32(r) << 16;
p++;
sscanf(in + p, "%d", &immediate);
binary += immediate & 0xffff;
binary += (0x5) << 26;
this->binary_code = binary;
binary code2str code();
```

```
    return 0;
}

int str2bin_j(char* in){
    bit32 target, binary = 0;
    sscanf(in, "%d", &target);
    binary += target & 0x3fffff;
    binary += (0x2) << 26;
    this->binary_code = binary;
    binary_code2str_code();

    return 0;
}

int str2bin_jal(char* in){
    bit32 target, binary = 0;
    sscanf(in, "%d", &target);
    binary += target & 0x3fffff;
    binary += (0x3) << 26;
    this->binary_code = binary;
    binary_code2str_code();

    return 0;
}

int str2bin_jr(char* in){
```

```
    int i = 0, r, p = 0;
    bit32 binary = 0;
    int immediate = 0;
    char *temp;
    temp = (char*)calloc(7, 1);
    while (in[p] != ' '){
        temp[i] = in[p];
        i++;
        p++;
        if (i > 6)break;
    }
    temp[i] = 0;
    if ((r = read_register(temp)) == -1)return -1;
    binary += bit32(r) << 21;
    binary += 0x8;
    this->binary_code = binary;
    binary_code2str_code();


    return 0;
}

void binary_code2str_code(){
    char *str, *reg_1, *reg_2, *reg_3;
    bit32 op, func, reg1, reg2, reg3, target, immediateu;
    int immediate;
    str = (char*)calloc(30, 1);
    reg_1 = (char*)calloc(7, 1);
    reg_2 = (char*)calloc(7, 1);
    reg_3 = (char*)calloc(7, 1);
    op = (this->binary_code) >> 26;
    func = (this->binary_code) & 0x3f;
    reg1 = ((this->binary_code) >> 21) & 0x1f;
    reg2 = ((this->binary_code) >> 16) & 0x1f;
    reg3 = ((this->binary_code) >> 11) & 0x1f;
    reg_1 = get_register_name(reg1);
    reg_2 = get_register_name(reg2);
    reg_3 = get_register_name(reg3);
    immediateu = (this->binary_code) & 0xffff;
    immediate = (immediateu & 0x8000) ? -(0x10000-immediateu) : (immediateu & 0x7fff);
    target = (this->binary_code) & 0x3ffffff;
    switch (op){
    case 0:
        switch (func){
        case 0x20:
            sprintf(str, "add %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x22:
            sprintf(str, "sub %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x2a:
            sprintf(str, "slt %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x24:
            sprintf(str, "and %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x25:
            sprintf(str, "or %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x27:
            sprintf(str, "nor %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x26:
```

```
            sprintf(str, "xor %s %s %s", reg_3, reg_1, reg_2); break;
        case 0x8:
            sprintf(str, "jr %s", reg_1); break;
        default:
            sprintf(str, "hot hot hot"); break;
        }break;
    case 0x8:
        sprintf(str, "addi %s %s %d", reg_2, reg_1, immediate); break;
    case 0xa:
        sprintf(str, "slti %s %s %d", reg_2, reg_1, immediate); break;
    case 0xc:
        sprintf(str, "andi %s %s %d", reg_2, reg_1, immediate); break;
    case 0xd:
        sprintf(str, "ori %s %s %d", reg_2, reg_1, immediate); break;
    case 0xe:
        sprintf(str, "xori %s %s %d", reg_2, reg_1, immediate); break;
    case 0x4:
        sprintf(str, "beq %s %s %d", reg_1, reg_2, immediate); break;
    case 0x5:
        sprintf(str, "bne %s %s %d", reg_1, reg_2, immediate); break;
    case 0x23:
        sprintf(str, "lw %s %d(%s)", reg_2, immediate, reg_1); break;
    case 0x2b:
        sprintf(str, "sw %s %d(%s)", reg_2, immediate, reg_1); break;
    case 0x20:
        sprintf(str, "lb %s %d(%s)", reg_2, immediate, reg_1); break;
    case 0x28:
        sprintf(str, "sb %s %d(%s)", reg_2, immediate, reg_1); break;
    case 0xf:
            sprintf(str, "lui %s %d", reg_2, immediate); break;
        case 0x2:
            sprintf(str, "j %d", target); break;
        case 0x3:
            sprintf(str, "jal %d", target); break;
        default:
            sprintf(str, "hot hot hot"); break;
        }
        this->str_code = str;
    }

    char* get_register_name(bit32 reg){
        char* temp;
        temp = (char*)calloc(7, 1);
        switch (reg){
        case 0:strcpy(temp, "$zero"); break;
        case 1:strcpy(temp, "$at"); break;
        case 2:strcpy(temp, "$v0"); break;
        case 3:strcpy(temp, "$v1"); break;
        case 4:strcpy(temp, "$a0"); break;
        case 5:strcpy(temp, "$a1"); break;
        case 6:strcpy(temp, "$a2"); break;
        case 7:strcpy(temp, "$a3"); break;
        case 8:strcpy(temp, "$t0"); break;
        case 9:strcpy(temp, "$t1"); break;
        case 10:strcpy(temp, "$t2"); break;
        case 11:strcpy(temp, "$t3"); break;
        case 12:strcpy(temp, "$t4"); break;
```

```
    case 13:strcpy(temp, "$t5"); break;
    case 14:strcpy(temp, "$t6"); break;
    case 15:strcpy(temp, "$t7"); break;
    case 16:strcpy(temp, "$s0"); break;
    case 17:strcpy(temp, "$s1"); break;
    case 18:strcpy(temp, "$s2"); break;
    case 19:strcpy(temp, "$s3"); break;
    case 20:strcpy(temp, "$s4"); break;
    case 21:strcpy(temp, "$s5"); break;
    case 22:strcpy(temp, "$s6"); break;
    case 23:strcpy(temp, "$s7"); break;
    case 24:strcpy(temp, "$t8"); break;
    case 25:strcpy(temp, "$t9"); break;
    case 26:strcpy(temp, "$k0"); break;
    case 27:strcpy(temp, "$k1"); break;
    case 28:strcpy(temp, "$gp"); break;
    case 29:strcpy(temp, "$sp"); break;
    case 30:strcpy(temp, "$fp"); break;
    case 31:strcpy(temp, "$ra"); break;
    default:strcpy(temp, "???"); break;
    }
    return temp;
}

int read_register(char* reg){
    if (!strcmp(reg, "$0"))return 0;
    else if (!strcmp(reg, "$1"))return 1;
    else if (!strcmp(reg, "$2"))return 2;
    else if (!strcmp(reg, "$3"))return 3;
    else if (!strcmp(reg, "$4"))return 4;
    else if (!strcmp(reg, "$5"))return 5;
    else if (!strcmp(reg, "$6"))return 6;
    else if (!strcmp(reg, "$7"))return 7;
    else if (!strcmp(reg, "$8"))return 8;
    else if (!strcmp(reg, "$9"))return 9;
    else if (!strcmp(reg, "$10"))return 10;
    else if (!strcmp(reg, "$11"))return 11;
    else if (!strcmp(reg, "$12"))return 12;
    else if (!strcmp(reg, "$13"))return 13;
    else if (!strcmp(reg, "$14"))return 14;
    else if (!strcmp(reg, "$15"))return 15;
    else if (!strcmp(reg, "$16"))return 16;
    else if (!strcmp(reg, "$17"))return 17;
    else if (!strcmp(reg, "$18"))return 18;
    else if (!strcmp(reg, "$19"))return 19;
    else if (!strcmp(reg, "$20"))return 20;
    else if (!strcmp(reg, "$21"))return 21;
    else if (!strcmp(reg, "$22"))return 22;
    else if (!strcmp(reg, "$23"))return 23;
    else if (!strcmp(reg, "$24"))return 24;
    else if (!strcmp(reg, "$25"))return 25;
    else if (!strcmp(reg, "$26"))return 26;
    else if (!strcmp(reg, "$27"))return 27;
    else if (!strcmp(reg, "$28"))return 28;
    else if (!strcmp(reg, "$29"))return 29;
    else if (!strcmp(reg, "$30"))return 30;
    else if (!strcmp(reg, "$31"))return 31;
```

```
        else if (!strcmp(reg, "$zero"))return 0;
        else if (!strcmp(reg, "$at"))return 1;
        else if (!strcmp(reg, "$v0"))return 2;
        else if (!strcmp(reg, "$v1"))return 3;
        else if (!strcmp(reg, "$a0"))return 4;
        else if (!strcmp(reg, "$a1"))return 5;
        else if (!strcmp(reg, "$a2"))return 6;
        else if (!strcmp(reg, "$a3"))return 7;
        else if (!strcmp(reg, "$t0"))return 8;
        else if (!strcmp(reg, "$t1"))return 9;
        else if (!strcmp(reg, "$t2"))return 10;
        else if (!strcmp(reg, "$t3"))return 11;
        else if (!strcmp(reg, "$t4"))return 12;
        else if (!strcmp(reg, "$t5"))return 13;
        else if (!strcmp(reg, "$t6"))return 14;
        else if (!strcmp(reg, "$t7"))return 15;
        else if (!strcmp(reg, "$s0"))return 16;
        else if (!strcmp(reg, "$s1"))return 17;
        else if (!strcmp(reg, "$s2"))return 18;
        else if (!strcmp(reg, "$s3"))return 19;
        else if (!strcmp(reg, "$s4"))return 20;
        else if (!strcmp(reg, "$s5"))return 21;
        else if (!strcmp(reg, "$s6"))return 22;
        else if (!strcmp(reg, "$s7"))return 23;
        else if (!strcmp(reg, "$t8"))return 24;
        else if (!strcmp(reg, "$t9"))return 25;
        else if (!strcmp(reg, "$k0"))return 26;
        else if (!strcmp(reg, "$k1"))return 27;
        else if (!strcmp(reg, "$gp"))return 28;
        else if (!strcmp(reg, "$sp"))return 29;
        else if (!strcmp(reg, "$fp"))return 30;
        else if (!strcmp(reg, "$ra"))return 31;
        else return -1;
    }

public:
    instruction(){
        binary_code = 0;
        str_code = 0;
    }

    bit32 get_binary_code(){
        return binary_code;
    }

    char* get_str_code(){
        return str_code;
    }

    char* C_str_binary_code(){
        char *temp;
        int value[8];
        temp = (char*)calloc(15, 1);
        value[0] = ((this->binary_code) >> 28) & 0xf;
        value[1] = ((this->binary_code) >> 24) & 0xf;
        value[2] = ((this->binary_code) >> 20) & 0xf;
        value[3] = ((this->binary_code) >> 16) & 0xf;
```

```
        value[4] = ((this->binary_code) >> 12) & 0xf;
        value[5] = ((this->binary_code) >> 8) & 0xf;
        value[6] = ((this->binary_code) >> 4) & 0xf;
        value[7] = (this->binary_code) & 0xf;
        sprintf(temp, "0x%x%x%x%x%x%x%x%x", value[0], value[1], value[2], value[3], value[4],
value[5], value[6], value[7]);
        return temp;
    }

    int input_instruction_str(char* in){
        int i = 0;
        char *temp;
        temp = (char*)calloc(6, 1);
        while (in[i] != ' '){
            temp[i] = in[i];
            if (temp[i] >= 'A'&&temp[i] <= 'Z')
                temp[i] += 32;
            i++;
            if (i > 5)break;
        }
        temp[i] = 0;
        if (!strcmp(temp, "add"))return str2bin_add(in + 4);
        else if (!strcmp(temp, "sub"))return str2bin_sub(in + 4);
        else if (!strcmp(temp, "slt"))return str2bin_slt(in + 4);
        else if (!strcmp(temp, "and"))return str2bin_and(in + 4);
        else if (!strcmp(temp, "or"))return str2bin_or(in + 3);
        else if (!strcmp(temp, "nor"))return str2bin_nor(in + 4);
        else if (!strcmp(temp, "xor"))return str2bin_xor(in + 4);
        else if (!strcmp(temp, "addi"))return str2bin_addi(in + 5);
        else if (!strcmp(temp, "slti"))return str2bin_slti(in + 5);
        else if (!strcmp(temp, "andi"))return str2bin_andi(in + 5);
        else if (!strcmp(temp, "ori"))return str2bin_ori(in + 4);
        else if (!strcmp(temp, "xori"))return str2bin_xori(in + 5);
        else if (!strcmp(temp, "lw"))return str2bin_lw(in + 3);
        else if (!strcmp(temp, "sw"))return str2bin_sw(in + 3);
        else if (!strcmp(temp, "lb"))return str2bin_lb(in + 3);
        else if (!strcmp(temp, "sb"))return str2bin_sb(in + 3);
        else if (!strcmp(temp, "lui"))return str2bin_lui(in + 4);
        else if (!strcmp(temp, "beq"))return str2bin_beq(in + 4);
        else if (!strcmp(temp, "bne"))return str2bin_bne(in + 4);
        else if (!strcmp(temp, "j"))return str2bin_j(in + 2);
        else if (!strcmp(temp, "jal"))return str2bin_jal(in + 4);
        else if (!strcmp(temp, "jr"))return str2bin_jr(in + 3);
        else return -1;
    }

    int input_instruction_binary(bit32 code){
        this->binary_code = code;
        binary_code2str_code();
        return 0;
    }
};

QString dodo(QString instring){
    instring = instring.simplified();
    string o;
    char*s=(char*)calloc(30, 1);
```

```
    QByteArray ba = instring.toLatin1();
    s=ba.data();
        instruction t;
        if (t.input_instruction_str(s) == -1){
            o +="Wrong input!";
        }
        else
        {
        string ast=t.C_str_binary_code();
        //string fn=t.get_str_code();;
        o=ast;
        }

    QString qo = QString::fromStdString(o);
    return qo;
}
QString odod(QString instring){
    string o;
    bool ok;
    bit32 s=instring.toUInt(&ok,16);
        instruction t;
        if (t.input_instruction_binary(s) == -1){
            o +="Wrong input!";
        }
        else
        {
        //string ast=t.C_str_binary_code();
        string fn=t.get_str_code();;
        o=fn;
        }

    QString qo = QString::fromStdString(o);
    return qo;
}
```