Y86-84 模拟器 PJ 实验报告

孙福特

18307130154

目录：

一．实验概述

本次 PJ 主要参照 CMU《深入理解计算机系统》第四章，完成 Y86-64 处理器指令集架构和五级流水线设计，通过暂停、转发等机制处理冒险和控制异常。我首先用 C++完成了底层逻辑的设计，用命令行窗口调试通过了给出的测试用例；然后用 Qt 完成图形界面的设计和展示。

二．代码说明

提交的学号文件夹中 code 目录下是我的源代码，其中：

pipe.cpp 是我的底层逻辑结构，与第一阶段提交的代码基本相同，主　要目的是更新并维护各寄存器、流水线寄存器、和内存等的值；

widget.h 是 Qt 中用到的管理窗口显示的代码，主要内容是定义了一个　窗口基类和其上的若干控件，它们有的用于展示实时数据，有的是控制模拟器 运行的热键。

widget.cpp 中实现了 widget 类的构造函数，当主函数执行到 widget　　对象的初始化语句时，该函数会生成最终的模拟器展示程序；同时，其中还定 义了若干信号和其槽函数，用于连接控制信号和模拟器更新。

Pj3.pro：Qt 工程管理文件。

三．模拟器使用方法

1.打开 y86sim 目录下的 y86.exe

2.在右上方选择栏选择测试用例，也可以选择“….”选项并在上面输入绝对路径，选择完成后点击 ok 进入测试模式，之后 ok 会变为 reset 以供之后更换测试用例；

3. step 按键作用是前进一个时钟周期;go 按键作用是让时钟按一定速度持续前进，这个速度的默认值是 1 帧，可以通过调整下方的滑条调整速度，最大帧数为 10;当自动运行模式开启时 go 按键会变为 pause，用来停止自动运行，停止后也可以再次启动；下面是一个 DEC/HEX 按键，用来在十进制和十六进制间切换数据显示方式；

4. 模拟器实时展示各流水线寄存器的值和 15 个程序寄存器的值，如果发生内存更新，读写文件错误等信息，会在右下方 message 框中显示；

5. 一个用例结束后可回到步骤 2 重新开始下一次测试；

四．实现的功能

1. 读.yo 文件，可重复

2. 模拟器的分步进行及演示

3. 模拟器的自动运行及演示，动态设置速度

4. 提供进制转换

5. 内存更新等信息显示

五．实现细节

后端部分：这部分代码在 code 文件夹中的 pipe.cpp 部分

我的底层代码主要包括：读写文件程序，流水线五阶段程序，暂停-转发-错误跳转的控制程序，它们中主要函数的作用如下：

读取文件并初始化的函数：

get_byte(int n,int i)：　用于从一行中第 i 个位置读 n 个 byte 保存到内存中的代码段

read_one_code()：　读一条代码

read_all_code()：　读一个.yo 文件中的所有代码

init()：对各个全局变量初始化，对跳转信息等也清空

一个时钟周期内的主要函数：

fetch（），decode（），execute（），memory（），write（）：五 个阶段的主体部分

gx_PC（）： 用于更新 PC

forwarding_stalling_control():流水线控制逻辑，用于处理暂停与转发

clock_rise_copy()：在每个时钟上升沿对流水线寄存器的值更新

runclock（int n）：对进行 n 个时钟周期的封装，具体使用基本都采用 n = 1

stall_1_clock(): 暂停一个周期，和 runclock（1）类似

circle（）：一个时钟周期的最终封装，用于前端中与 step 信号的槽连接

用来展示的函数：（它们在图形界面中并没有用到）

show（）：展示各个流水线寄存器的值

show_reg（）： 展示各寄存器的值

show_mem(): 展示内存中的值

show_code（）：展示保存的代码

前端部分：

run_prepare()：每次添加新的用例时初始化的函数。

下面均为主窗口 Widget 类中函数：

setlabel（）和 setbutton（）设置按钮和标签样式位置的函数

void set_label()： 更新图形界面中显示内容

槽函数：

void myslot_step()：step 按钮的槽函数，会使后端调用一次circle（），运行一个时钟周期

void myslot_ok()：ok 按钮的槽函数

void myslot_go()：go 按钮的槽函数，会开始计数器，并将时钟触发连接到 step 的槽函数，自动模拟时钟演进

void myslot_pause()：暂停 go 的函数，停止计数器

void myslot_DECHEX()：改变显示进制的函数

void changeV()：滑条的槽函数，改变时钟速度

我将几个重要步骤用一些例子来展示。

1.读入文件我采取的是逐行读入，定位代码到内存段的方式，不支持一　　行多条语句，这是一个只得改进之处；

```
void read_all_code(){
    while(!f.eof() && f.good()){
        f.getline(line , 1000, '\n');
        //cout << line << endl;
        if(line[5] != ':' || line[7] == ' ')continue;
        int a = line[3] - '0';
        if(a > 9 && a < 0)                    △ overla
            a += '0' - 'a' + 10;
        int b = line[4] - '0';
        if(b > 9 || b < 0)
            b += '0' - 'a' + 10;
        PC = 16 * a + b;
        read_one_code();
    }
}
```

2.GUI 界面布局我采取的是写代码的传统编程方式，未采用图形化编程

```
void Widget::setlabel(QLabel& l,int x,int y, QString str,int f){
    l.setParent(this);
    l.move(x,y);
    l.setText(str);
    if(f == 1){l.setFont(font); l.setPalette(pe);}
    else if(f == 2){l.setFont(font2);l.setPalette(pe2);}
    else if(f == 3){l.setFont(font3);l.setPalette(pe3);}
    else if(f == 4){l.setFont(font4);l.setPalette(pe4);}
}
```

```
    font3.setPointSize(15);
    font4.setPointSize(20);
    pe.setColor(QPalette::WindowText, Qt::yellow);
    pe2.setColor(QPalette::WindowText, Qt::red);
    pe3.setColor(QPalette::WindowText, Qt::white);
    pe4.setColor(QPalette::WindowText, Qt::gray);
```

```
    setlabel(n_Dl[0],150,380,"D_state",1);
    setlabel(n_Dl[1],250,380,"D_icode",1);
    setlabel(n_Dl[2],350,380,"D_ifun",1);
    setlabel(n_Dl[3],450,380,"D_rA",1);
    setlabel(n_Dl[4],550,380,"D_rB",1);
    setlabel(n_Dl[5],650,380,"D_valC",1);
```

3. 模拟事件触发方式我采取的是标准的槽与信号的链接方法

```
connect(&step, &QPushButton::released, this, &Widget::myslot_step);
connect(&OK, &QPushButton::released, this, &Widget::myslot_ok);
connect(&DEC, &QPushButton::released, this, &Widget::myslot_DECHEX);
connect(&slider,&QSlider::valueChanged, this, &Widget::changeV);
```

```
void Widget::myslot_step(){
    if(lef <= 3 && lef > 0){
        d_state = 0;
        run_clock(1);
        lef--;
    }
    else if(!lef){
        l3.setText("END");
        return;
    }
    else if(!circle())lef = 3;
    set_label();
}
```

4. 计数器的触发我借助了 Qt 中类 Qtimer 来完成

```
timer = new QTimer(this);
connect(timer, &QTimer::timeout, this, &Widget::myslot_step);
```

```
void Widget::myslot_go(){
    timer->start(1000/v);
    go.setText("pause");
    disconnect(&go, &QPushButton::released, this, &Widget::myslot_go);
    connect(&go, &QPushButton::released, this, &Widget::myslot_pause);
}
```

六. 待改进之处

1. 上文已提及的文件读入方式
2. 实现插入断点的分步调试功能
3. 界面美化

七. 参考资料

很多文献和资料对我项目的设计提供了帮助，包括但不限于：

1. 机械工业出版社《计算机系统基础》（英文原书《Computer Systems》Randal E. Bryant ， David R. O’Hallaron）龚奕利译

2. https://www.cnblogs.com/wyuzm/p/9594274.html

https://blog.csdn.net/CatCherry/article/details/98770902

https://blog.csdn.net/qq_33308135/article/details/82895525

https://blog.csdn.net/yang6464158/article/details/38050747

https://www.cnblogs.com/wanghuixi/p/9521717.html

Qt 中 Qlabel，QpushBotton，QlineEdit，Qcomobox 等空间的使用方法

3. https://blog.csdn.net/qq_36880027/article/details/98765716

https://blog.csdn.net/qq_39054069/article/details/96481902

动态链接库与可执行文件的生成