计算机组成原理实验报告

13349051 劳嘉辉 计科一班

## 实验内容

使用C++或者C语言生成一个程序。这个程序可以执行lab1中lc2k生成的机器码的内容。实际上就是生成一个行为模拟器。

## 实验目的

1. 更好地了解LC2K汇编语言
2. 更好地了解机器码是如何执行的
3. 更好地了解到机器码执行的过程中内存memory和寄存器reg内容的变化。

## 代码显示

1、定义内存寄存器结构体

typedef struct stateStruct {

int pc;

int mem[NUMMEMORY];

int reg[NUMREGS];

int numMemory;

} stateType;

这个stateStruct模拟一个机器内部的容器。Pc指的就是program control 程序控制，用于判断目前执行到哪一个步骤。而mem[]则是模拟一个连续的内存空间，里面装的是instruction或者是offset。Reg[]则是模拟一个寄存器空间，里面存放的是模拟寄存器的值。Men[]和reg[]在一开始的时候都会被memset()函数清空为0。而numMemory则是用来该程序需要多少的空间。

2、打印内存寄存器结构体内部情况的函数

void printState(stateType \*statePtr)

{

int i;

printf("\n@@@\nstate:\n");

printf("\tpc %d\n", statePtr->pc);

printf("\tmemory:\n");

for (i = 0; i<statePtr->numMemory; i++) {

printf("\t\tmem[ %d ] %d\n", i, statePtr->mem[i]);

}

printf("\tregisters:\n");

for (i = 0; i<NUMREGS; i++) {

printf("\t\treg[ %d ] %d\n", i, statePtr->reg[i]);

}

printf("end state\n");

}

该函数主要是利用struct中的元素来展示该结构体内部有什么。然后用printf()函数展示出来。

3、signed数转换为unsigned数

Int convertNum(int num)

{

/\* convert a 16-bit number into a 32-bit Linux integer \*/

if (num & (1 << 15)) {

num -= (1 << 16);

}

return(num);

}

这是老师给的函数，这种做法我以前没有用过，确实很巧妙。这里是把二的十五次方与num比较，其实就是比较两个数字的首个字符，如果都是1的话，说明num其实一个负数，那么就把num减去二的十六次方从而得到相应的负数如果num的首个数字是0，那么说明这个num是一个正数，那么就没有转换的必要了。

4、把十进制的数转换成二进制的机器码

string processnum(int num)

{

string temp="00000000000000000000000000000000";

int index = 31;

while (num)

{

if (num % 2 == 1)

temp[index] = '1';

index--;

num /= 2;

}

return temp;

}

运用欧几里得辗转相除法就可以得到相应的二进制机器码了。

5、把offset的16位二进制数转换为十进制数

Int offsetnum(string num)

{

int sum = 0;

for (int i = 15; i >=0;i--)

if (num[i] == '1')

sum += pow(2.0, 15-i);

return convertNum(sum);

}

6、难点指令——beq

if (command.substr(7, 3) == "100")

{

printState(&state);

int rega = 0;

int regb = 0;

int offset = 0;

rega = regnum(command.substr(10, 3));

regb = regnum(command.substr(13, 3));

offset = offsetnum(command.substr(16, 16));

if (state.reg[rega] == state.reg[regb])

{

int newnumMemory = state.pc + 1 + offset;

rewind(filePtr);

while (newnumMemory--)

fgets(line, MAXLINELENGTH, filePtr);

state.pc += offset;

}

instruction++;

}//beq

本次实验中比较有难度的指令就是beq指令了，因为这个指令产生了分支将打乱了原有的指令顺序。这里的话我先算出了offset的值，然后算出新的pc值，然后使用rewind()函数重新读取文件头部，然后在fgets()函数读到相应的内存值，其他的指令较为简单，因此在这里不再赘述。

## 实验案例

实验案例较多，因此另外在文件夹里面显示。

## 实验心得

这次实验我又学到了很多C语言的新做法，例如<<这种逻辑偏移的方法在之前学习c++是没有涉及到的，这为以后的编程也带来了很多的便利。更重要的是我进一步复习了机器码指令。从add、nand、lw、sw、beq、jalr、halt、noop每个指令都有不同的操作，而在编码的过程中我又复习了一遍操作，而printstate这个函数则让我更加深刻地了解到在机器码执行的过程中内存寄存器究竟发生了什么。本次实验令我受益确实良多。