|  |
| --- |
| Федеральное агентство связи Государственное образовательное учреждение высшего профессионального образования «Сибирский государственный университет телекоммуникаций и информатики»  Кафедра вычислительных систем |
| Курсовая работа по дисциплине «ЭВМ и периферийные устройства» |
|  |
|  |
| Выполнил: Студент группы ИП-13 Рафальский Д.В. |
|  |

|  |
| --- |
|  |

Проверил:  
Майданов Ю. С.

Новосибирск - 2013

Содержание:

1.Постановка задачи 2  
2. Выполнение работы 3  
2.1. Модель центрального процессора 3  
2.2. Транслятор с языка Simple Assembler 3  
3. Код программы 4  
4. Пример работы Simple Computer 12  
5. Вывод 13

1. Постановка задачи.

В период изучения курса «ЭВМ и периферийные устройства» в нашей курсовой работе была поставлена задача разработать программную модель простейшей вычислительной машины Simple Computer, транслятора с языка Simple Assembler.

2. Выполнение работы.

2.1. Модель центрального процессора.

Описаны две функции для выполнения программ моделью Simple Computer:  
- int ALU (int command, int operand) - реализует алгоритм работы арифметико-логического устройства. Если при выполнении функции возникла ошибка, которая не позволяет дальше выполнять программу, то функция возвращает -1, иначе 0;

- int CU (void) – обеспечивает работу устройства управления. Является функцией обратного вызова, вызываемая системным таймером или по нажатию клавиши «t» в случае установленного флага пошаговой работы. Производит чтение из массива – модели оперативной памяти со смещением, хранящемся в регистре - указателе команд InstructionCounter и декодирует полученное значение. В результате декодирования возможны следующие исходы:  
- устройство управления самостоятельно обрабатывает инструкцию, если команда не требует арифметических вычислений. Возвращает «0».  
- устройство управления передает обработку команды в АЛУ, если команда требует арифметических вычислений над операндами. К этой группе команд относятся команды, выполняющие арифметические и логические действия. Возвращает «0».  
- в случае поступления некорректной команды (с недопустимым операционным кодом или операндом ) устройство управления останавливает работу SimpleComputer и возвращает «-1».

2.2. Транслятор с языка Simple Assembler.

Задачей транслятора является преобразование текстового файла, содержащего программный код на языке SimpleAssembler, в бинарный файл, содержащий дамп памяти SimpleComputer (дамп содержит программу в машинном коде).  
  
Пример программы, вычисляющей остаток от деления:  
  
00 READ 09  
01 READ 10  
02 LOAD 09  
03 SUB 10  
04 STORE 09  
05 LOAD 09  
06 JNS 02  
07 WRITE 09  
08 HALT 00  
09 = +0000  
10 = +0000

3. Код программы

#include "console.h"

#include <sys/time.h>

#include <signal.h>

#include <iostream>

using namespace std;

int selt,F;

struct itimerval nval, oval;

int alu(int command, int operand){

int t,tt;

switch(command){

case(30):

sc\_memoryGet(operand, &t);

if(accumulator + t>0x7FFF){

sc\_regSet(T, 1);

sc\_regSet(P, 1);

}

accumulator+=t;

break;

case(31):

sc\_memoryGet(operand, &t);

if(accumulator - t<0){

sc\_regSet(T, 1);

sc\_regSet(P, 1);

}

accumulator -= t;

break;

case(32):

sc\_memoryGet(operand, &t);

if(t==0){

sc\_regSet(T, 1);

sc\_regSet(O, 1);

}

accumulator /= t;

break;

case(33):

sc\_memoryGet(operand, &t);

if(accumulator \* t>0x7FFF){

sc\_regSet(T, 1);

sc\_regSet(P, 1);

}

accumulator \*= t;

break;

case(51):

t = ~accumulator;

sc\_memorySet(operand, t);

break;

case(52):

sc\_memoryGet(operand, &t);

accumulator = accumulator&t;

break;

case(53):

sc\_memoryGet(operand, &t);

accumulator = accumulator^t;

break;

case(55):

if(accumulator>0)

instructionCounter=operand-1;

}

return 0;

}

int cu(){

int command, operand;

sc\_commandDecode(sc\_memory[instructionCounter], &command, &operand);

if(command>76||command<10){

sc\_regSet(T, 1);

sc\_regSet(E, 1);

}

if(operand>99||operand<0){

sc\_regSet(M, 1);

sc\_regSet(T, 1);

}

int t;

switch(command){

case(10):

F=1;

sc\_regSet(T, 1);

cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1); printf("Print the value of the memory cell ¹%d - ", operand);

scanf("%x", &t);

sc\_memorySet(operand, t);

cout<<"\E[?25l";

mt\_gotoXY(27,1);

rk\_mytermregime(0,0,1,0,1);

sc\_regSet(T, 0);

F=0;

break;

case(11):

F=1;

sc\_regSet(T, 1);

cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

sc\_memoryGet(operand, &t);

mt\_gotoXY(27,1); printf("The value of the memory cell ¹%d = %d", operand, t);

cout<<"\E[?25l";

mt\_gotoXY(27,1);

rk\_mytermregime(0,0,1,0,1);

sc\_regSet(T, 0);

F=0;

break;

case(17):

accumulator = operand;

case(20):

sc\_memoryGet(operand, &accumulator);

break;

case(21):

sc\_memorySet(operand, accumulator);

break;

case(30):

case(31):

case(32):

case(33):

alu(command, operand);

break;

case(40):

instructionCounter = operand;

break;

case(41):

if(accumulator<0){

instructionCounter= operand;

}

break;

case(42):

if(accumulator==0){

instructionCounter= operand;

}

break;

case(43):

sc\_regSet(T, 1);

break;

case(51):case(52):case(53):case(54):case(55):case(56):case(57):case(58):case(59):

case(60):case(61):case(62):case(63):case(64):case(65):case(66):case(67):case(68):case(69):

case(70):case(71):case(72):case(73):case(74):case(75):case(76):

alu(command, operand);

break;

default:

sc\_regSet(T, 1);

sc\_regSet(E, 1);

cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<command<<endl;

break;

}

return 0;

}

void timer\_on(int signo) {

F=0;

int t;

sc\_regGet (T,&t);

if(t==0) {

if(instructionCounter>99) {

sc\_regSet(M, 1);

sc\_regSet(T, 1);

}

else {

cu();

Draw(ind);

ind=++instructionCounter;

}

}

setitimer(ITIMER\_REAL,&nval, &oval);

}

void timer\_off(int signo) {

setitimer(ITIMER\_REAL,0,0);

}

void move() {

keys k;

char str[10];

int isIMPIGNORE;

int cell=0;

sc\_memoryInit();

sc\_regInit();

sc\_regSet(T,1);

struct itimerval nval, oval;

nval.it\_interval.tv\_sec=0;

nval.it\_interval.tv\_usec=10000;

nval.it\_value.tv\_sec=0;

nval.it\_value.tv\_usec=10000;

signal (SIGALRM, timer\_on);

signal (SIGUSR1, timer\_off);

cout<<"\E[?25l";

draw(0);

while(1) {

rk\_mytermregime(0,0,1,0,1);

sc\_regGet (T,&isIMPIGNORE);

if (isIMPIGNORE==0)

setitimer(ITIMER\_REAL,&nval, &oval);

rk\_readkey(k);

if (k==r) {

if (isIMPIGNORE==1) {

sc\_regSet(T,0);

selt=ind;

Draw(ind);

continue;

}

}

if (isIMPIGNORE == 1) {

switch(k) {

case k\_right: if((ind+1)%10!=0) ind++; Draw(ind); break;

case k\_left: if(ind%10!=0) ind--; Draw(ind); break;

case down: if(ind<90) ind+=10; Draw(ind); break;

case up: if(ind>9) ind-=10; Draw(ind); break;

case l: cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1);

cout<<"Enter filename: ";

cin>>str;

sc\_memoryLoad(str);

cout<<"\E[?25l";

mt\_gotoXY(27,1); cout<<" ";

Draw(ind); break;

case s: cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1);

cout<<"Enter filename: ";

cin>>str;

sc\_memorySave(str);

cout<<"\E[?25l";

mt\_gotoXY(27,1); cout<<" ";

Draw(ind); break;

case f5:

accumulator=sc\_memory[ind];

Draw(ind); break;

case f6:

cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1);

cout<<"Enter new value of instructionCounter: ";

scanf("%X",&cell);

instructionCounter=cell;

cout<<"\E[?25l";

mt\_gotoXY(27,1); cout<<" ";

Draw(ind); break;

case enter: cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1);

cout<<"Enter new value: ";

scanf("%X",&cell);

if(cell>0xFFFF) {

mt\_gotoXY(27,1); cout<<" ";

mt\_gotoXY(27,1); cout<<"Error, overflow!";

}

else sc\_memorySet(ind,cell);

cout<<"\E[?25l";

Draw(ind); break;

case t:

ind=instructionCounter;

cu();

instructionCounter++;

Draw(ind);

sc\_regSet(T,1);

break;

case 5:

sc\_memoryInit();

sc\_regInit();

sc\_regSet(T,1);

Draw(ind);

break;

default: mt\_gotoXY(27,1); break;

}

if(k==13) {mt\_gotoXY(27,1); break;}

}

}

cout<<"\E[?12;25h";

rk\_mytermregime(1,0,50,1,1);

}

int main() {

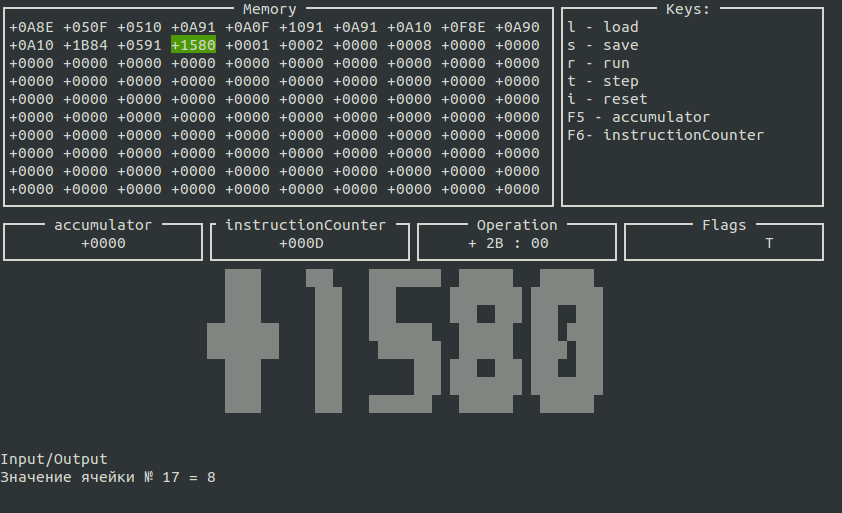
move();

return 0;

}

4. Пример работы Simple Computer.





5. Вывод.

В ходе выполнения данной курсовой работы был разобран принцип работы вычислительной машины. Была разработана простейшая модель вычислительной машины, включающая в себя:  
- память  
- устройства ввода-вывода  
- устройство управления  
- арифметико-логическое устройство  
Реализован транслятор для языка Simple Assembler.