```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
using namespace std;

unsigned int text[2 * 1024 / 4];
int registers[32];
int pc;

void add(int c, int a, int b) {
registers[c] = registers[a] + registers[b];
}

void andfunc(int c, int a, int b) {
registers[c] = registers[a] && registers[b];
}

void orfunc(int c, int a, int b) {
registers[c] = registers[a] || registers[b];
}

void sub(int c, int a, int b) {
registers[c] = registers[a] - registers[b];
}

void slt(int c, int a, int b) {
if (registers[a] < registers[b])
registers[c] = 1;
else
registers[c] = 0;
}

void beq(int a, int b, int c) {
if (registers[a] == registers[b])
pc += ((c & 0xFFFFF) - 1);
}

void bne(int a, int b, int c) {
if (registers[a] != registers[b])
pc += (c & 0xFFFFF) - 1;
}

void jump(int c) {
```

```
pc = (pc & 0xF0000000) | (c & 0xFFFF);
}

void syscall() {
int v0 = registers[2];
switch (v0) {
case 1:
printf("%d", registers[4]);
break;
case 2:
scanf_s("%d", registers[2]);
break;
case 3:
exit(1);
break;
}
}

void parse(int instruction) {
pc += 1;
registers[0] = 0;
int opcode = instruction && 0xFC000000;
int rs = instruction && 0x3E00000;
int rt = instruction && 0x1F0000;
int rd = instruction && 0xF800;
int funct = instruction && 0x3F;
short imm = instruction && 0xFFFF;
int address = instruction && 0x3FFFFFF;

switch (opcode) {
case 0x0:
switch (funct)
{
case 0x20:
add(rd, rs, rt);
break;
case 0x22:
sub(rd, rs, rt);
break;
case 0x24:
andfunc(rd, rs, rt);
break;
case 0x25:
orfunc(rd, rs, rt);
break;
case 42:
```

```
slt(rd, rs, rt);
break;
case 0xC:
syscall();
break;
}
break;
case 0x2:
jump(address);
break;
case 0x4:
beq(rs, rt, imm);
break;
case 0x5:
bne(rs, rt, imm);
break;
default:
cout << "There is not a valid instruction" << endl;
break;
}
}

void readFile(string filename) {
string number;
ifstream myfile(filename.c_str());
vector<string> entireFile(40000);
if (myfile.is_open()) {

int i = 0;
while (!myfile.eof()) {
getline(myfile, number);
cout << "line is " << number << endl;
entireFile[i] = number;
i++;
}
myfile.close();
}

int size;
unsigned int j;
for (j = 0; j<entireFile.size(); j++) {

if (entireFile[j] == "DATA SEGMENT") {
size = j;
cout << "Found data segment at " << size << endl;
break;
```

```
}
cout << "String is: " << entireFile[j] << endl;
scanf_s(entireFile[j].c_str(), "%x", &text[j]);
cout << "text at " << dec << j << " is " << hex << text[j] << endl;
}
cout << "Split String Loop" << endl;
unsigned int k;
for (k = size + 1; k < entireFile.size(); k++) {
string first = entireFile[k];
if (first.empty())
break;
cout << "Split string: " << first << endl;
string firstStr = first.substr(0, 20);
string secondStr = first.substr(21, 10);
}
}

int main(int argc, char* argv[]) {
pc = 0;
string fileName;
fileName = "123.o";
int mode;
mode = 0;
readFile(fileName);
if (mode == 0) {
cout << "Run to completion mode------" << endl;
while (true) {
parse(text[pc]);
}
}
else if (mode == 1) {
cout << "Single step mode------" << endl;
while (1) {
string input;
cin >> input;
if (input.substr(0, input.length()) == "p_all") {
for (int i = 0; i<32; i++) {
cout << "register " << dec << i << ": " << hex
<< registers[i] << endl;
}
}
else if (input.at(0) == 'p') {
int regN;
scanf_s(input.substr(2, input.size() - 2).c_str(), "%d", &regN);
cout << "register " << dec << regN << ": " << hex
<< registers[regN] << endl;
```

```
}
if (input.at(0) == 's') {
int skip;
scanf_s(input.substr(2, input.size() - 2).c_str(), "%d", &skip);
int i;
int instr;
for (i = 0; i < skip; i++) {
instr = text[pc];
cout << "Instruction: " << hex << instr << endl;
parse(instr);
}
}
}
}
}
return 0;
}
```