#include <cstdlib>

#include <fstream>

#include <iomanip>

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct instruction {

string inst;

int f = 0;

int b = 0;

string op = "";

int target = 0;

};

struct RsAdd {

string op[4] = {};

string f[4] = {};

string b[4] = {};

int executeT[4] = {};

string addBufferRs;

int findpos() {

for (int i = 0; i < 3; i++) {

if (op[i] == "")

return i;

}

return -1;

};

int answerBuffer = 0;

};

struct RsMul {

string op[3] = {};

string f[3] = {};

string b[3] = {};

int executeT[3] = {};

string mulBufferRs;

int findpos() {

for (int i = 0; i < 2; i++) {

if (op[i] == "")

return i;

}

return -1;

};

int answerBuffer = 0;

};

struct Register {

string RSpresent[5] = {"RS1", "RS2", "RS3", "RS4", "RS5"};

string RF[5] = {};

string RAT[5] = {};

int issueindex = 0;

};

Register Reg;

RsAdd RADD;

RsMul RMUL;

string getope(string s) {

string buf = "";

// bool breakflag= false;

for (int i = 2; i < s.size(); i++) {

if (s[i] == ' ') {

break;

}

buf = buf + s[i];

}

return buf;

};

void updateResult(string answer, string buf) {

for (int i = 0; i < 3; i++) { // ADDreg

if (RADD.f[i] == buf)

RADD.f[i] = answer;

if (RADD.b[i] == buf)

RADD.b[i] = answer;

} // 原本想用replace寫但想說算了

for (int i = 0; i < 2; i++) { // MILreg

if (RMUL.f[i] == buf)

RMUL.f[i] = answer;

if (RMUL.b[i] == buf)

RMUL.b[i] = answer;

}

for (int i = 0; i < 5; i++) { // RAT update

if (Reg.RAT[i] == buf) {

Reg.RAT[i] = "";

Reg.RF[i] = answer;

}

}

};

string replacerat(int index) {

index = index - 1;

if (Reg.RAT[index] == "")

return Reg.RF[index];

else

return Reg.RAT[index];

};

int AddreadyDispatch() {

for (int i = 0; i < 3; i++) {

if (RADD.executeT[i] == 0) {

if (RADD.f[i] != "" && RADD.b[i] != "") {

if (isdigit(RADD.f[i][0]) && isdigit(RADD.b[i][0])) {

return i;

}

}

}

}

return -1;

};

int MulreadyDispatch() {

for (int i = 0; i < 2; i++) {

if (RMUL.executeT[i] == 0) {

if (RMUL.f[i] != "" && RMUL.b[i] != "") {

if (isdigit(RMUL.f[i][0]) && isdigit(RMUL.b[i][0])) {

return i;

}

}

}

}

return -1;

};

int CAL(string op, string f, string b) {

int ff, bb;

if (op == "+") {

ff = stoi(f);

bb = stoi(b);

return ff + bb;

} else if (op == "-") {

ff = stoi(f);

bb = stoi(b);

return ff - bb;

} else if (op == "\*") {

ff = stoi(f);

bb = stoi(b);

return ff \* bb;

} else { // div

ff = stoi(f);

bb = stoi(b);

return ff / bb;

}

};

bool breakornot() {

// bool b=false;

for (int i = 0; i < 3; i++) {

if (RADD.op[i] != "")

return false;

}

for (int i = 0; i < 2; i++) {

if (RMUL.op[i] != "") {

return false;

}

}

return true;

};

int main() {

for (int i = 0; i < 5; i++) { // RF init

Reg.RF[i] = to\_string(2 \* i);

// cout << Reg.RF[i] << endl;

}

vector<instruction> AllInst;

ifstream readf("input.txt", ios::in);

if (!readf.is\_open())

cout << "Open failed" << endl;

else {

string str;

while (getline(readf, str)) { // read and prase

int j = 0;

instruction buffer;

buffer.inst = str;

while (str[j] != ' ') {

buffer.op = buffer.op + str[j];

j++;

}

j++;

while (str[j] != ',') {

j++;

}

buffer.target = (str[j - 1]) - '0';

j++;

while (str[j] != ',') {

j++;

}

buffer.f = (str[j - 1]) - '0';

while (str[j] != '\0') {

j++;

}

buffer.b = (str[j - 1]) - '0';

AllInst.push\_back(buffer);

}

readf.close();

}

// for (int i = 0; i < AllInst.size(); i++) {

// cout << AllInst[i].op << endl

// << AllInst[i].target << endl

// << AllInst[i].f << endl

// << AllInst[i].b << endl;

// }

int cycle = 1;

bool changed = false;

bool done = false;

int R[32] = {0};

while (!done) {

// write back

if (RADD.executeT[3] == 0 && RADD.op[3] != "") {

changed = true;

updateResult(to\_string(RADD.answerBuffer), RADD.addBufferRs);

for (int j = 0; j < 3; j++) { // clean RADD

if (Reg.RSpresent[j] == RADD.addBufferRs) {

RADD.op[j] = "";

RADD.f[j] = "";

RADD.b[j] = "";

}

}

// 檢查這裡需不需要clean buffer

RADD.op[3] = "";

}

if (RMUL.executeT[2] == 0 && RMUL.op[2] != "") {

changed = true;

updateResult(to\_string(RMUL.answerBuffer), RMUL.mulBufferRs);

for (int j = 3; j < 5; j++) { // clean RADD

if (Reg.RSpresent[j] == RMUL.mulBufferRs) {

RMUL.op[j - 3] = "";

RMUL.f[j - 3] = "";

RMUL.b[j - 3] = "";

}

}

RMUL.op[2] = "";

}

// issue

if (Reg.issueindex < AllInst.size()) {

instruction instbuffer;

instbuffer = AllInst[Reg.issueindex];

if (instbuffer.op == "ADD" || instbuffer.op == "ADDI") {

int pos = RADD.findpos();

if (pos > -1) {

RADD.op[pos] = "+";

RADD.f[pos] = replacerat(instbuffer.f);

if (instbuffer.op == "ADDI")

RADD.b[pos] = to\_string(instbuffer.b);

else

RADD.b[pos] = replacerat(instbuffer.b);

RADD.executeT[pos] = 1;

Reg.RAT[instbuffer.target - 1] = Reg.RSpresent[pos];

Reg.issueindex++;

}

} else if (instbuffer.op == "SUB") {

int pos = RADD.findpos();

if (pos > -1) {

RADD.op[pos] = "-";

RADD.f[pos] = replacerat(instbuffer.f);

RADD.b[pos] = replacerat(instbuffer.b);

RADD.executeT[pos] = 1;

Reg.RAT[instbuffer.target - 1] = Reg.RSpresent[pos];

Reg.issueindex++;

}

} else if (instbuffer.op == "MUL") {

int pos = RMUL.findpos();

if (pos > -1) {

RMUL.op[pos] = "\*";

RMUL.f[pos] = replacerat(instbuffer.f);

RMUL.b[pos] = replacerat(instbuffer.b);

RMUL.executeT[pos] = 1;

Reg.RAT[instbuffer.target - 1] = Reg.RSpresent[pos + 3];

Reg.issueindex++;

}

} else { // DIV

int pos = RMUL.findpos();

if (pos > -1) {

RMUL.op[pos] = "/";

RMUL.f[pos] = replacerat(instbuffer.f);

RMUL.b[pos] = replacerat(instbuffer.b);

RMUL.executeT[pos] = 1;

Reg.RAT[instbuffer.target - 1] = Reg.RSpresent[pos + 3];

Reg.issueindex++;

}

}

changed = true;

}

// dispatch

int pos = AddreadyDispatch();

if (pos != -1 && RADD.op[3] == "") {

RADD.op[3] = RADD.op[pos];

RADD.f[3] = RADD.f[pos];

RADD.b[3] = RADD.b[pos];

RADD.executeT[3] = 2;

RADD.addBufferRs = Reg.RSpresent[pos];

RADD.answerBuffer = CAL(RADD.op[3], RADD.f[3], RADD.b[3]);

changed = true;

}

pos = MulreadyDispatch();

if (pos != -1 && RMUL.op[2] == "") {

RMUL.op[2] = RMUL.op[pos];

RMUL.f[2] = RMUL.f[pos];

RMUL.b[2] = RMUL.b[pos];

RMUL.executeT[2] = 4;

RMUL.mulBufferRs = Reg.RSpresent[pos + 3];

RMUL.answerBuffer = CAL(RMUL.op[2], RMUL.f[2], RMUL.b[2]);

changed = true;

}

// run --> - executeT

for (int i = 0; i < 3; i++) {

if (RADD.op[i] != "" && RADD.executeT[i] != 0)

RADD.executeT[i]--;

}

if (RADD.executeT[3] != 0) {

RADD.executeT[3]--;

}

for (int i = 0; i < 2; i++) {

if (RMUL.op[i] != "" && RMUL.executeT[i] != 0)

RMUL.executeT[i]--;

}

if (RMUL.executeT[2] != 0) {

RMUL.executeT[2]--;

}

// show

if (changed) {

cout << "Cycle: " << cycle << endl;

cout << endl;

cout << " \_ RF \_\_" << endl;

cout << " F1 |" << setw(4) << Reg.RF[0] << " | " << endl;

cout << " F2 |" << setw(4) << Reg.RF[1] << " | " << endl;

cout << " F3 |" << setw(4) << Reg.RF[2] << " | " << endl;

cout << " F4 |" << setw(4) << Reg.RF[3] << " | " << endl;

cout << " F5 |" << setw(4) << Reg.RF[4] << " | " << endl;

cout << " -------" << endl;

cout << endl;

cout << " \_ RAT \_\_" << endl;

cout << " F1 |" << setw(5) << Reg.RAT[0] << " | " << endl;

cout << " F2 |" << setw(5) << Reg.RAT[1] << " | " << endl;

cout << " F3 |" << setw(5) << Reg.RAT[2] << " | " << endl;

cout << " F4 |" << setw(5) << Reg.RAT[3] << " | " << endl;

cout << " F5 |" << setw(5) << Reg.RAT[4] << " | " << endl;

cout << " --------" << endl;

cout << endl;

cout << " \_ RS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " RS1 |" << setw(5) << RADD.op[0] << " |" << setw(5) << RADD.f[0]

<< " |" << setw(5) << RADD.b[0] << " | " << endl;

cout << " RS2 |" << setw(5) << RADD.op[1] << " |" << setw(5)

<< RADD.f[1] << " |" << setw(5) << RADD.b[1]

<< " | " << endl;

cout << " RS3 |" << setw(5) << RADD.op[2] << " |" << setw(5)

<< RADD.f[2] << " |" << setw(5) << RADD.b[2]

<< " | " << endl;

cout << " ----------------------" << endl;

if (RADD.op[3]=="") {

cout << "BUFFER: empty" << endl;

} else {

cout << "BUFFER: (" << RADD.addBufferRs << ")"

<< RADD.f[3] << RADD.op[3]

<< RADD.b[3] << endl;

}

cout << endl;

cout << " \_ RS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " RS4 |" << setw(5) << RMUL.op[0] << " |" << setw(5)

<< RMUL.f[0] << " |" << setw(5) << RMUL.b[0]

<< " | " << endl;

cout << " RS5 |" << setw(5) << RMUL.op[1] << " |" << setw(5)

<< RMUL.f[1] << " |" << setw(5) << RMUL.b[1]

<< " | " << endl;

cout << " ----------------------" << endl;

if (RMUL.op[2] == "") {

cout << "BUFFER: empty" << endl;

} else {

cout << "BUFFER: (" << RMUL.mulBufferRs<< ")"

<< RMUL.f[2] << RMUL.op[2]

<< RMUL.b[2] << endl;

}

cout << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

// break

if (breakornot()) {

break;

}

cycle++;

}

}