#include <iostream>

#include <vector>

#include <string>

#include <map>

#include <fstream>

#include <sstream>

#include <cstring>

#include <iomanip>

#include <cmath>

using namespace std;

class RSdata {

public:

char op;

string rs1, rs2;

bool used = 0, able = 0;

int cycleCount = 0;

string index, rd;

int ans;

};

class inst {

public:

string op, rd, rs1, rs2;

};

stringstream ss;

vector <inst> insts;

map <string, int> RF;

map <string, string> RAT;

map <string, char> operators = { { "ADD",'+' } , { "ADDI",'+' }, { "SUB",'-' }, { "MUL",'\*' }, { "DIV",'/' } };

vector <RSdata> RS1, RS2;

RSdata emp;

RSdata\* buffer1;

RSdata\* buffer2;

RSdata\* buffer3;

int cycle1, cycle2, cycle3;

int RS1count = 0, RS2count = 0;

bool change = 0;

bool unit1 = 0, unit2 = 0, unit3 = 0;

int n = 0;

int cycle = 0;

int busycount1 = 0, busycount2 = 0;

void input() {

cout << "Please input cycle for ADD, ADDI, SUB: ";

cin >> cycle1;

cout << "Please input cycle for MUL: ";

cin >> cycle2;

cout << "Please input cycle for DIV: ";

cin >> cycle3;

}

void init() {

RF["F1"] = 0;

RF["F2"] = 2;

RF["F3"] = 4;

RF["F4"] = 6;

RF["F5"] = 8;

RS1.resize(4);

RS2.resize(3);

}

void read() {

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

if (!inFile) {

cout << "File could not be opened!\n";

system("pause");

exit(1);

}

while (!inFile.eof()) {

ss.str("");

ss.clear();

inst i;

string buffer;

getline(inFile, buffer);

ss << buffer;

ss >> i.op;

ss >> i.rd;

ss >> i.rs1;

ss >> i.rs2;

i.rd.erase(i.rd.end() - 1);

i.rs1.erase(i.rs1.end() - 1);

insts.push\_back(i);

}

}

void issue() {

if (n < insts.size()) {

if (insts[n].op == "ADD" || insts[n].op == "ADDI" || insts[n].op == "SUB") {

if (RS1count < 3) {

for (int i = 1; i <= 3; ++i) {

if (!RS1[i].used) {

RS1[i].op = operators[insts[n].op];

if (RAT[insts[n].rs1] == "") {

RS1[i].rs1 = to\_string(RF[insts[n].rs1]);

}

else {

RS1[i].rs1 = RAT[insts[n].rs1];

}

if (isdigit(insts[n].rs2[0])) {

RS1[i].rs2 = insts[n].rs2;

}

else {

if (RAT[insts[n].rs2] == "") {

RS1[i].rs2 = to\_string(RF[insts[n].rs2]);

}

else {

RS1[i].rs2 = RAT[insts[n].rs2];

}

}

RAT[insts[n].rd] = "RS" + to\_string(i);

RS1[i].index = "RS" + to\_string(i);

RS1[i].rd = insts[n].rd;

RS1[i].used = 1;

++RS1count;

++n;

++busycount1;

change = 1;

break;

}

}

}

}

else {

if (RS2count < 2) {

for (int i = 1; i <= 2; ++i) {

if (!RS2[i].used) {

RS2[i].op = operators[insts[n].op];

if (RAT[insts[n].rs1] == "") {

RS2[i].rs1 = to\_string(RF[insts[n].rs1]);

}

else {

RS2[i].rs1 = RAT[insts[n].rs1];

}

if (RAT[insts[n].rs2] == "") {

RS2[i].rs2 = to\_string(RF[insts[n].rs2]);

}

else {

RS2[i].rs2 = RAT[insts[n].rs2];

}

RAT[insts[n].rd] = "RS" + to\_string(i + 3);

RS2[i].index = "RS" + to\_string(i + 3);

RS2[i].rd = insts[n].rd;

RS2[i].used = 1;

++RS2count;

++n;

++busycount2;

change = 1;

break;

}

}

}

}

}

}

void dispatch() {

if (!unit1) {

for (int i = 1; i <= 3; ++i) {

if (RS1[i].used && isdigit(RS1[i].rs1[0]) && isdigit(RS1[i].rs2[0]) && RS1[i].able) {

buffer1 = &RS1[i];

unit1 = 1;

++buffer1->cycleCount;

change = 1;

break;

}

}

}

else {

++buffer1->cycleCount;

}

if (!unit2) {

for (int i = 1; i <= 2; ++i) {

if (RS2[i].used && isdigit(RS2[i].rs1[0]) && isdigit(RS2[i].rs2[0]) && RS2[i].able && RS2[i].op == '\*') {

buffer2 = &RS2[i];

unit2 = 1;

++buffer2->cycleCount;

change = 1;

break;

}

}

}

else {

++buffer2->cycleCount;

}

if (!unit3) {

for (int i = 1; i <= 2; ++i) {

if (RS2[i].used && isdigit(RS2[i].rs1[0]) && isdigit(RS2[i].rs2[0]) && RS2[i].able && RS2[i].op == '/') {

buffer3 = &RS2[i];

unit3 = 1;

++buffer3->cycleCount;

change = 1;

break;

}

}

}

else {

++buffer3->cycleCount;

}

}

void writeback() {

for (int i = 1; i <= 3; ++i) {

if (RS1[i].used) {

RS1[i].able = 1;

}

}

for (int i = 1; i <= 2; ++i) {

if (RS2[i].used) {

RS2[i].able = 1;

}

}

if (unit1) {

if (buffer1->cycleCount == cycle1 + 1) {

if (buffer1->op == '+') {

buffer1->ans = stoi(buffer1->rs1) + stoi(buffer1->rs2);

}

else if (buffer1->op == '-') {

buffer1->ans = stoi(buffer1->rs1) - stoi(buffer1->rs2);

}

for (int i = 1; i <= 3; ++i) {

if (RS1[i].rs1 == buffer1->index) {

RS1[i].rs1 = to\_string(buffer1->ans);

}

if (RS1[i].rs2 == buffer1->index) {

RS1[i].rs2 = to\_string(buffer1->ans);

}

}

for (int i = 1; i <= 2; ++i) {

if (RS2[i].rs1 == buffer1->index) {

RS2[i].rs1 = to\_string(buffer1->ans);

}

if (RS2[i].rs2 == buffer1->index) {

RS2[i].rs2 = to\_string(buffer1->ans);

}

}

RAT.erase(buffer1->index);

RF[buffer1->rd] = buffer1->ans;

\*buffer1 = emp;

buffer1 = nullptr;

--RS1count;

unit1 = 0;

--busycount1;

change = 1;

}

}

if (unit2) {

if (buffer2->cycleCount == cycle2 + 1) {

buffer2->ans = stoi(buffer2->rs1) \* stoi(buffer2->rs2);

for (int i = 1; i <= 3; ++i) {

if (RS1[i].rs1 == buffer2->index) {

RS1[i].rs1 = to\_string(buffer2->ans);

}

if (RS1[i].rs2 == buffer2->index) {

RS1[i].rs2 = to\_string(buffer2->ans);

}

}

for (int i = 1; i <= 2; ++i) {

if (RS2[i].rs1 == buffer2->index) {

RS2[i].rs1 = to\_string(buffer2->ans);

}

if (RS2[i].rs2 == buffer2->index) {

RS2[i].rs2 = to\_string(buffer2->ans);

}

}

RAT.erase(buffer2->index);

RF[buffer2->rd] = buffer2->ans;

\*buffer2 = emp;

buffer2 = nullptr;

--RS2count;

unit2 = 0;

--busycount2;

change = 1;

}

}

if (unit3) {

if (buffer3->cycleCount == cycle3 + 1) {

buffer3->ans = stoi(buffer3->rs1) / stoi(buffer3->rs2);

for (int i = 1; i <= 3; ++i) {

if (RS1[i].rs1 == buffer3->index) {

RS1[i].rs1 = to\_string(buffer3->ans);

}

if (RS1[i].rs2 == buffer3->index) {

RS1[i].rs2 = to\_string(buffer3->ans);

}

}

for (int i = 1; i <= 2; ++i) {

if (RS2[i].rs1 == buffer3->index) {

RS2[i].rs1 = to\_string(buffer3->ans);

}

if (RS2[i].rs2 == buffer3->index) {

RS2[i].rs2 = to\_string(buffer3->ans);

}

}

RAT.erase(buffer3->index);

RF[buffer3->rd] = buffer3->ans;

\*buffer3 = emp;

buffer3 = nullptr;

--RS2count;

unit3 = 0;

--busycount2;

change = 1;

}

}

}

void output(ostream& outFile) {

outFile << endl << "Cycle: " << cycle << endl;

outFile << " \_ RF \_\_\n";

outFile << " F1 | " << RF["F1"] << " |\n";

outFile << " F2 | " << RF["F2"] << " |\n";

outFile << " F3 | " << RF["F3"] << " |\n";

outFile << " F4 | " << RF["F4"] << " |\n";

outFile << " F5 | " << RF["F5"] << " |\n";

outFile << " -------\n";

outFile << " \_ RAT \_\_\n";

outFile << " F1 | " << RAT["F1"] << " |\n";

outFile << " F2 | " << RAT["F2"] << " |\n";

outFile << " F3 | " << RAT["F3"] << " |\n";

outFile << " F4 | " << RAT["F4"] << " |\n";

outFile << " F5 | " << RAT["F5"] << " |\n";

outFile << " --------\n";

outFile << " \_ RS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

outFile << " RS1 | " << RS1[1].op << " | " << RS1[1].rs1 << " | " << RS1[1].rs2 << " |\n";

outFile << " RS2 | " << RS1[2].op << " | " << RS1[2].rs1 << " | " << RS1[2].rs2 << " |\n";

outFile << " RS3 | " << RS1[3].op << " | " << RS1[3].rs1 << " | " << RS1[3].rs2 << " |\n";

outFile << " ----------------------\n";

outFile << "BUFFER: ";

if (buffer1 == nullptr) {

outFile << "empty\n";

}

else {

outFile << "(" << buffer1->index << ") " << buffer1->rs1 << " " << buffer1->op << " " << buffer1->rs2 << endl;

}

outFile << endl << endl << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

outFile << " RS4 | " << RS2[0].op << " | " << RS2[0].rs1 << " | " << RS2[0].rs2 << " |\n";

outFile << " RS5 | " << RS2[1].op << " | " << RS2[1].rs1 << " | " << RS2[1].rs2 << " |\n";

outFile << " ----------------------\n";

outFile << "BUFFER: ";

if (buffer2 == nullptr) {

outFile << "empty\n";

}

else {

outFile << "(" << buffer2->index << ") " << buffer2->rs1 << " " << buffer2->op << " " << buffer2->rs2 << endl;

}

outFile << "BUFFER: ";

if (buffer3 == nullptr) {

outFile << "empty\n";

}

else {

outFile << "(" << buffer3->index << ") " << buffer3->rs1 << " " << buffer3->op << " " << buffer3->rs2 << endl;

}

}

void exec() {

ofstream outFile("output.txt", ios::out);

if (!outFile) {

cout << "File could not be opened!\n";

system("pause");

exit(1);

}

while (true) {

change = 0;

issue();

dispatch();

writeback();

if (change == 1) {

output(cout);

output(outFile);

}

if (busycount1 == 0 && busycount2 == 0) {

RAT.clear();

break;

}

++cycle;

}

}

int main() {

input();

init();

read();

exec();

}