Computer Architecture

Assignment 1
Branch Predictors

Names

- Aya Ashraf (2)
- Sohayla Mohammed (32)

Tournament Predictor

Code

main

```
int main() {
    Driver *tournment = new Driver();
    string input = "";
    ifstream in ("input.txt");
    ofstream outfile ("output.txt");
    int lineNum = 0;

    if (in.is_open())
    {

        while (getline(in, input))
        {
            lineNum++;
            bool correct_P = input.substr(1) == "n"? false : true;
            string index = input.substr(0,1);
            bool local = tournment->getLocalPrediction(index, correct_P);
            bool global = tournment->getGlobalPrediction(correct_P);
            string selector = tournment->getSelectorPrediction(index, correct_P, local, global);
            bool prediction = tournment->getCorrectPrediction(local, global, selector);
            tournment->updateCorrectness(local, global, prediction, correct_P);
            outfile << (writeData(lineNum, index, correct_P,local, global, prediction, selector ));
            outfile << endl;
        }

        outfile << "Statistics :" << endl;
        outfile << "#of calls = " << to_string(lineNum) << endl;
        outfile << "#of branches = " << "9" << endl;
        outfile << "#of correct global = " << tournment->getCorrectGlobal()<<endl;
        outfile << "#of correct local = " << tournment->getcorrectLocal()<<endl;
        outfile << "#of correct Tournament prediction = " << tournment->getCorrectTournment();

        outfile.close();
        in.close();
    }

    return 0;
}
```

```
#include <iostream>
#include <fstream>
#include "Driver.h"

string writeData(string basic_string, bool p, bool local, bool global, bool prediction, string basicString);

using namespace std;
string writeData(int lineNum, string index, bool p, bool local, bool global, bool prediction, string selector) {
    string correct = p? "t" : "n";
    string local_p = local? "t" : "n";
    string global_p = global? "t" : "n";
    string final_p = prediction? "t" : "n";

    string output =  to_string(lineNum) + " - br#" + index + " - correct_p " + correct + +
            " - Actual " + final_p + " - Selector "+ selector + " (local " + local_p +
               " - global " + global_p + ")";

    return output;
}
```

Driver.h

```
//
// Created by sohayla on 13/03/19.
//

#ifndef TOURNMENT_PREDICTOR_DRIVER_H
#define TOURNMENT_PREDICTOR_DRIVER_H

#include <map>
#include "LocalPredictor.h"
#include "Selector.h"
#include "GlobalPredictor.h"

using namespace std;
class Driver {
private:
    map<string, Selector*> selectorMap;
    map<string, LocalPredictor*> localMap;
    map<string, GlobalPredictor*> globalMap;
    int correctLocal, correctGlobal, correctTournment;
    string local, global;
    string globalIndex;
    void updateGlobalIndex(bool prediction);

public:
    Driver();
    string getSelectorPrediction(string index, bool correctPrediction, bool localPrediction, bool globalPrediction);
    bool getGlobalPrediction(bool correctPrediction);
    bool getLocalPrediction(string index, bool correctPrediction);
    bool getCorrectPrediction(bool local_p, bool global_p, string selector_p);
    void updateCorrectness(bool local_p, bool global_p, bool final, bool correct);
    int getcorrectLocal();
    int getCorrectGlobal();
    int getCorrectTournment();
};


#endif //TOURNMENT_PREDICTOR_DRIVER_H
```

Driver.cpp

```
//
// Created by sohayla on 13/03/19.
//

#include "Driver.h"
Driver::Driver() {
    correctGlobal = 0;
    correctLocal = 0;
    correctTournment = 0;
    local = "l";
    global = "g";
    globalIndex = "nnnn"; //Based on 4 previous prediction
}

void Driver::updateGlobalIndex(bool prediction) {
    if(prediction) {
        globalIndex = globalIndex.substr(1) + "t";
    } else {
        globalIndex = globalIndex.substr(1) + "n";
    }
}

bool Driver::getGlobalPrediction(bool correctPrediction) {
    GlobalPredictor *globalPredictor;
    if(globalMap.count(globalIndex)) {
        globalPredictor = globalMap[globalIndex];
    } else {
        globalPredictor = new GlobalPredictor(globalIndex);
        globalMap.insert(pair<string, GlobalPredictor*>(globalIndex, globalPredictor));
    }
    bool prediction = globalPredictor->getGlobalPrediction();
    globalPredictor->updateState(correctPrediction);
    updateGlobalIndex(prediction);
    return  prediction;
}



bool Driver::getLocalPrediction(string index, bool correctPrediction) {
    LocalPredictor *localPredictor;
    if(localMap.count(index)) {
        localPredictor = localMap[index];
    } else {
        localPredictor = new LocalPredictor(index);
        localMap.insert(pair<string, LocalPredictor*>(index, localPredictor));
    }
    bool prediction = localPredictor->getLocalPrediction();
    localPredictor->updateState(correctPrediction);
    return  prediction;
}

string Driver::getSelectorPrediction(string index, bool correctPrediction, bool localPrediction, bool globalPrediction) {
    Selector *selector;
    if(selectorMap.count(index)) {
        selector = selectorMap[index];
    } else {
        selector = new Selector(index);
        selectorMap.insert(pair<string, Selector*>(index, selector));
    }
    string prediction = selector->getSelection();
    selector->updateSelectionState(localPrediction, globalPrediction, correctPrediction);
    return  prediction;
}

bool Driver::getCorrectPrediction(bool local_p, bool global_p, string selector_p) {
    if(local == selector_p) {
        return local_p;
    } else {
        return global_p;
    }
}
```

```
void Driver::updateCorrectness(bool local_p, bool global_p, bool final, bool correct) {
    if((correct && local_p) || (!correct && !local_p)) {
        correctLocal++;
    }
    if((correct && local_p) || (!correct && !global_p)) {
        correctGlobal++;
    }
    if((correct && final) || (!correct && !final)) {
        correctTournment++;
    }
}

int Driver::getcorrectLocal() {
    return  correctLocal;
}
int Driver::getCorrectGlobal() {
    return  correctGlobal;
}
int Driver::getCorrectTournment() {
    return correctTournment;
}
```

LocalPredictor.h

```
//
// Created by sohayla on 13/03/19.
//

#ifndef TOURNMENT_PREDICTOR_LOCALPREDICTOR_H
#define TOURNMENT_PREDICTOR_LOCALPREDICTOR_H

#include <string>
#include "State.h"

using namespace std;
class LocalPredictor {
private:
    string index;
    State *currentState;
public:
    LocalPredictor(string index);
    bool getLocalPrediction();
    void updateTaken();
    void updateNotTaken();
    string getIndex();
    void updateState(bool branchResolution);

};


#endif //TOURNMENT_PREDICTOR_LOCALPREDICTOR_H
```

LocalPredictor.cpp

```
//
// Created by sohayla on 13/03/19.
//

#include "LocalPredictor.h"
LocalPredictor::LocalPredictor(string index) {
    this->index = index;
    this->currentState = new State();
}

bool LocalPredictor::getLocalPrediction() {
    int counterValue = currentState->getState();
    if(counterValue > 1) {
        return true;
    }

    return false;
}

void LocalPredictor::updateNotTaken() {
    this->currentState->decrementState();
}

void LocalPredictor::updateTaken() {
    this->currentState->incrementState();
}


void LocalPredictor::updateState(bool branchResolution) {
    if(branchResolution) {
        this->updateTaken();
    } else {
        this->updateNotTaken();
    }
}

string LocalPredictor::getIndex() {
    return this->index;
}
```

GlobalPredictor.h

```
//
// Created by sohayla on 13/03/19.
//

#ifndef TOURNMENT_PREDICTOR_GLOBALPREDICTOR_H
#define TOURNMENT_PREDICTOR_GLOBALPREDICTOR_H

#include <string>
#include "State.h"

using namespace std;

class GlobalPredictor {
private:
    string index;
    State *currentState;
public:
    GlobalPredictor(string index);
    bool getGlobalPrediction();
    void updateTaken();
    void updateNotTaken();
    string getIndex();
    void updateState(bool branchResolution);


};


#endif //TOURNMENT_PREDICTOR_GLOBALPREDICTOR_H
```

GlobalPredictor.cpp

```
//
// Created by sohayla on 13/03/19.
//

#include "GlobalPredictor.h"

GlobalPredictor::GlobalPredictor(string index) {
    this->index = index;
    this->currentState = new State();
}

bool GlobalPredictor::getGlobalPrediction() {
    int counterValue = currentState->getState();
    if(counterValue > 1) {
        return true;
    }

    return false;
}

void GlobalPredictor::updateNotTaken() {
    this->currentState->decrementState();
}

void GlobalPredictor::updateTaken() {
    this->currentState->incrementState();
}


void GlobalPredictor::updateState(bool branchResolution) {
    if(branchResolution) {
        this->updateTaken();
    } else {
        this->updateNotTaken();
    }
}

string GlobalPredictor::getIndex() {
    return this->index;
}
```

Selector.h

```
//
// Created by sohayla on 13/03/19.
//

#ifndef TOURNMENT_PREDICTOR_SELECTOR_H
#define TOURNMENT_PREDICTOR_SELECTOR_H

#include <string>
#include <map>
#include "State.h"

using namespace std;
class Selector {
private:
    string index;
    State *currentState;
public:
    Selector(string index);
    string getSelection();
    void updateSelectionState(bool local, bool global, bool actual);
    string getIndex();
};


#endif //TOURNMENT_PREDICTOR_SELECTOR_H
```

Selector.cpp

```
//
// Created by sohayla on 13/03/19.
//

#include "Selector.h"
Selector::Selector(string index) {
    this->index = index;
     currentState = new State();
}

void Selector::updateSelectionState(bool local, bool global, bool actual) {
    if((local && global) || (!local && !global)) {
        //do nothing
    } else if ((actual && global) || (!actual && !global)) {
        this->currentState->incrementState();
    } else {
        this->currentState->decrementState();
    }
}

string Selector::getSelection() {
    int counterValue = currentState->getState();
    if(counterValue > 1) {
        return "g";
    }

    return "l";
}

string Selector::getIndex() {
    return  this->index;
}
```

State.h

```
//
// Created by sohayla on 13/03/19.
//

#ifndef TOURNMENT_PREDICTOR_STATE_H
#define TOURNMENT_PREDICTOR_STATE_H


class State {
private:
    int state;
public:
    State();
    void incrementState();
    void decrementState();
    int getState();

};


#endif //TOURNMENT_PREDICTOR_STATE_H
```

State.cpp

```
//
// Created by sohayla on 13/03/19.
//

#include "State.h"
State::State() {
    this->state = 0;
}

void State::incrementState() {
    if(this->state < 3) {
        this->state++;
    }
}


void State::decrementState() {
    if(this->state > 0) {
        this->state--;
    }
}

int State::getState() {
    return  this->state;
}
```

Test case :

Input File :

```
2t
1t
7n
8t
3n
4t
5t
8t
3n
4n
5t
8n
9t
5t
8n
9t
0t
2t	3n
1t	4n
7n	5t
8t	8n
3n	9t
4t	0t
5t	2t
8t	1t
3n	7n
4n	8t
5t	3n
8n	4t
9t	5t
0t	8t
2t	3n
1t	4n
7n	5t
8t	8n
3n	9t
4t	0t
5t	2t
8t	1t
```

Output

Form :

line# - branch_index - correctPrediction - TournmentPrediction - Selector (LocalPred - globalPred)

```
1 - br#2 - correct_p t - Actual n - Selector l (local n - global n)
2 - br#1 - correct_p t - Actual n - Selector l (local n - global n)
3 - br#7 - correct_p n - Actual n - Selector l (local n - global t)
4 - br#8 - correct_p t - Actual n - Selector l (local n - global n)
5 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
6 - br#4 - correct_p t - Actual n - Selector l (local n - global n)
7 - br#5 - correct_p t - Actual n - Selector l (local n - global n)
8 - br#8 - correct_p t - Actual n - Selector l (local n - global n)
9 - br#3 - correct_p n - Actual n - Selector l (local n - global t)
10 - br#4 - correct_p n - Actual n - Selector l (local n - global n)
11 - br#5 - correct_p t - Actual n - Selector l (local n - global n)
12 - br#8 - correct_p n - Actual t - Selector l (local t - global n)
13 - br#9 - correct_p t - Actual n - Selector l (local n - global n)
14 - br#5 - correct_p t - Actual t - Selector l (local t - global n)
15 - br#8 - correct_p n - Actual n - Selector l (local n - global t)
16 - br#9 - correct_p t - Actual n - Selector l (local n - global n)
17 - br#0 - correct_p t - Actual n - Selector l (local n - global n)
18 - br#2 - correct_p t - Actual n - Selector l (local n - global n)
19 - br#1 - correct_p t - Actual n - Selector l (local n - global t)
20 - br#7 - correct_p n - Actual n - Selector l (local n - global n)
21 - br#8 - correct_p t - Actual n - Selector l (local n - global t)
22 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
23 - br#4 - correct_p t - Actual n - Selector l (local n - global n)
24 - br#5 - correct_p t - Actual t - Selector l (local t - global n)
25 - br#8 - correct_p t - Actual n - Selector l (local n - global t)
26 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
27 - br#4 - correct_p n - Actual n - Selector l (local n - global t)
28 - br#5 - correct_p t - Actual t - Selector l (local t - global n)
29 - br#8 - correct_p n - Actual n - Selector g (local t - global n)
30 - br#9 - correct_p t - Actual t - Selector l (local t - global t)
31 - br#0 - correct_p t - Actual n - Selector l (local n - global n)
32 - br#2 - correct_p t - Actual t - Selector l (local t - global t)
33 - br#1 - correct_p t - Actual t - Selector l (local t - global n)
34 - br#7 - correct_p n - Actual n - Selector l (local n - global n)
35 - br#8 - correct_p t - Actual t - Selector g (local n - global t)
36 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
37 - br#4 - correct_p t - Actual n - Selector l (local n - global t)
38 - br#5 - correct_p t - Actual t - Selector l (local t - global t)
39 - br#8 - correct_p t - Actual n - Selector g (local t - global n)
```

```
40 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
41 - br#4 - correct_p n - Actual n - Selector l (local n - global n)
42 - br#5 - correct_p t - Actual t - Selector l (local t - global t)
43 - br#8 - correct_p n - Actual n - Selector g (local t - global n)
44 - br#9 - correct_p t - Actual t - Selector l (local t - global t)
45 - br#0 - correct_p t - Actual t - Selector l (local t - global t)
46 - br#2 - correct_p t - Actual t - Selector l (local t - global n)
47 - br#1 - correct_p t - Actual t - Selector l (local t - global n)
48 - br#7 - correct_p n - Actual n - Selector l (local n - global n)
49 - br#8 - correct_p t - Actual t - Selector g (local t - global t)
50 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
51 - br#4 - correct_p t - Actual n - Selector l (local n - global t)
52 - br#5 - correct_p t - Actual t - Selector l (local t - global t)
53 - br#8 - correct_p t - Actual t - Selector g (local t - global t)
54 - br#3 - correct_p n - Actual n - Selector l (local n - global n)
55 - br#4 - correct_p n - Actual n - Selector g (local n - global n)
56 - br#5 - correct_p t - Actual t - Selector l (local t - global n)
57 - br#8 - correct_p n - Actual t - Selector g (local t - global t)
58 - br#9 - correct_p t - Actual t - Selector l (local t - global n)
59 - br#0 - correct_p t - Actual t - Selector l (local t - global t)
60 - br#2 - correct_p t - Actual t - Selector l (local t - global t)
61 - br#1 - correct_p t - Actual t - Selector l (local t - global t)
Statistics :
#of calls = 61
#of branches = 9
#of correct global = 37
#of correct local = 38
#of correct Tournament prediction = 40
```

Accuracy% = (40/61) * 100% = 65.574%

Correlated Predictor

```java
public class Main {
    public static void main(String[] args) {
         CorrelatingBranches b = new CorrelatingBranches();
         b.Correlarte();
         System.out.println("Accuracy = " + b.getAccuracy());
        }
}
```

```java
import java.util.ArrayList;

/*
 * this is an implementation of Correlating branch predictor according to 1
 * level prediction based on the states of the last branch if it taken or not
 * if Taken => predict from the branch Target Buffer "Taken"
 * if NotTaken => predict from the branch Target Buffer "NotTaken"
 *  1-bit predictor with 1-bit correlation
 *  condition true => branch NotTaken
 *  condition false => branch taken*/
public class CorrelatingBranches {
    private final int NUMBER_OF_BRANCHES = 2;
    private final int  NUMBER_OF_EXECUTION = 10;
    private ArrayList<Boolean> branchNotTaken;
    private ArrayList<Boolean> branchTaken;
    private boolean lastState = false;
    private ArrayList<Boolean> correctAction;
    private int rightprediction;
    private Random random;

    public CorrelatingBranches() {
        random = new Random();
        branchNotTaken = generator();
        System.out.println("predict Not Taken : " + branchNotTaken.toString());
        branchTaken = generator();
        System.out.println("predict Taken     : " + branchTaken.toString());
        correctAction = generator();
        System.out.println("True Value        : " + correctAction.toString());

        rightprediction = 0;
    }
```

```
void Correlarte() {
    for(int i = 0 ; i < NUMBER_OF_EXECUTION; i++) {
        int randomBranch = random.nextInt(NUMBER_OF_BRANCHES);
        boolean trueValue = correctAction.get(randomBranch);
        boolean predictedValue;
        if(lastState) {
            predictedValue = branchNotTaken.get(randomBranch);
            if (predictedValue == trueValue) {
                rightprediction ++;
            }else {
                branchNotTaken.remove(randomBranch);
                branchNotTaken.add(randomBranch, trueValue);
                lastState = !lastState;
            }

        }else {
            predictedValue = branchTaken.get(randomBranch);
            if (predictedValue == trueValue) {
                rightprediction ++;
            }else {
                branchTaken.remove(randomBranch);
                branchTaken.add(randomBranch, trueValue);
                lastState = !lastState;
            }
        }
        System.out.println("Number Of the Branch (" + randomBranch + ")" + " TValue (" + trueValue + ") PValue (" + predictedValue + ")");
        System.out.println("predict Not Taken :" + branchNotTaken.toString());
        System.out.println("predict     Taken :" + branchTaken.toString());

    }
}
```

```
private ArrayList<Boolean> generator() {
    ArrayList<Boolean> arr = new ArrayList<Boolean>();
    for(int i = 0 ; i < NUMBER_OF_BRANCHES ; i++) {
        arr.add(random.nextBoolean());
    }
    return arr;
}
public float getAccuracy() {
    return (float)rightprediction /(float) NUMBER_OF_EXECUTION;
}
```

Sample Run :-

```
predict Not Taken : [false, false]
predict Taken     : [false, true]
True Value        : [true, false]
Number Of the Branch (0) TValue (true) PValue (false)
predict Not Taken :[false, false]
predict     Taken :[true, true]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[false, false]
predict     Taken :[true, true]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[false, false]
predict     Taken :[true, true]
Number Of the Branch (0) TValue (true) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, true]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, true]
Number Of the Branch (1) TValue (false) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Accuracy = 0.7
```

```
predict Not Taken : [true, false]
predict Taken     : [true, true]
True Value        : [true, false]
Number Of the Branch (1) TValue (false) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (1) TValue (false) PValue (false)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Number Of the Branch (0) TValue (true) PValue (true)
predict Not Taken :[true, false]
predict     Taken :[true, false]
Accuracy = 0.9
```