COL 216 Assignment-2

Harshdeep 2020CS10346 & Pravin Kumar 2020CS10369

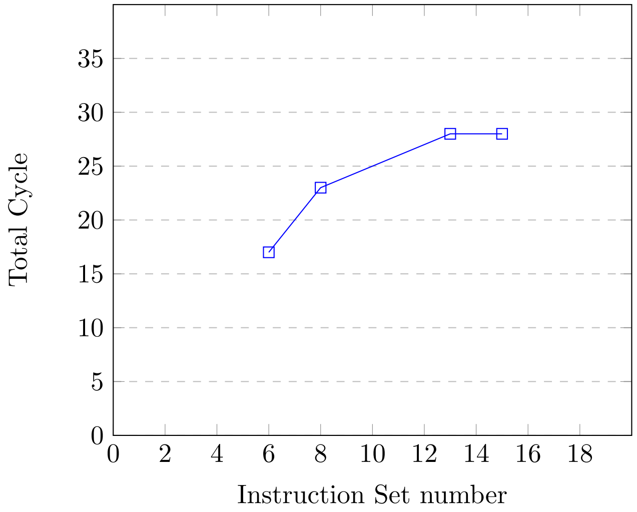
April 2023

A. 5 Stage

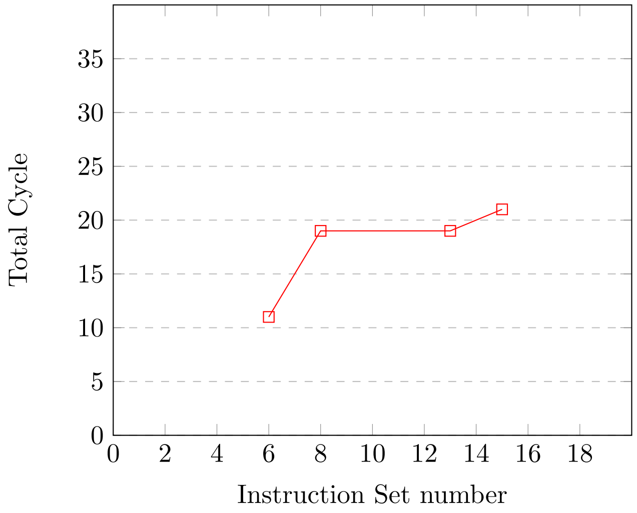
Cycle comparison between without bypassing and with bypassing

5stage				
Testcase	Instruction Count	Without Bypassing	With Bypassing	Improvement
public_test1	6	17	11	35.29%
public_test2	8	23	19	17.39%
public_test3	15	28	21	25%
public_test4	13	28	19	32.14%
sample	54	89	88	1.12%

5stage Without Bypassing



5stage With Bypassing



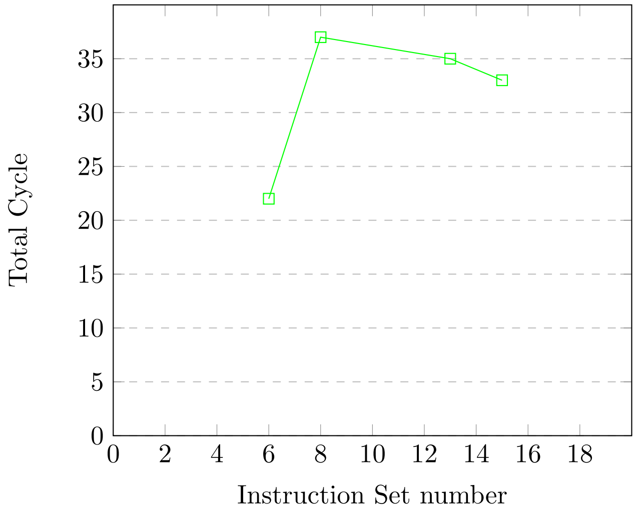
We found out that No. of cycles remains constant if increased instructions uses latch.

B. 7-9 Stage

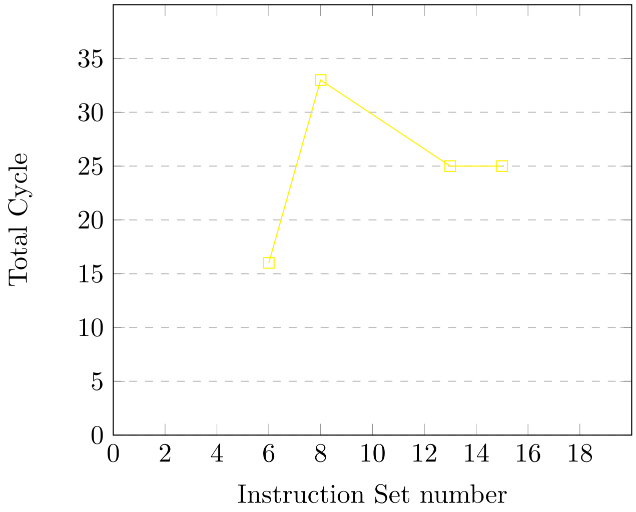
Cycle comparison between without bypassing and with bypassing

7-9stage				
Testcase	Instruction Count	Without Bypassing	With Bypassing	Improvement
public_test1	6	22	16	27.27%
public_test2	8	37	33	10.81%
public_test3	15	33	25	24.24%
public_test4	13	35	25	28.57%
sample	54	162	161	0.61%

7-9stage Without Bypassing



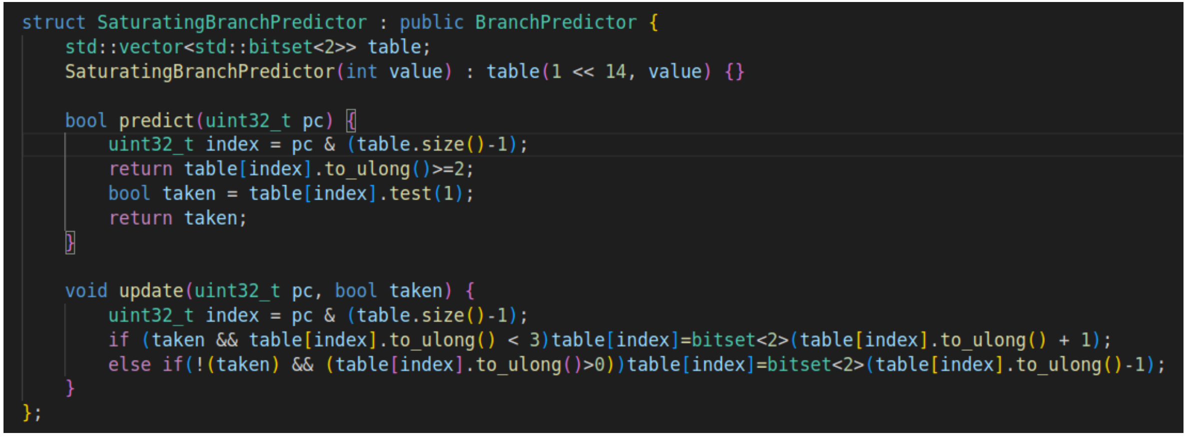
7-9stage With Bypassing



C. Branch Predictor

2-bit saturating branch predictor

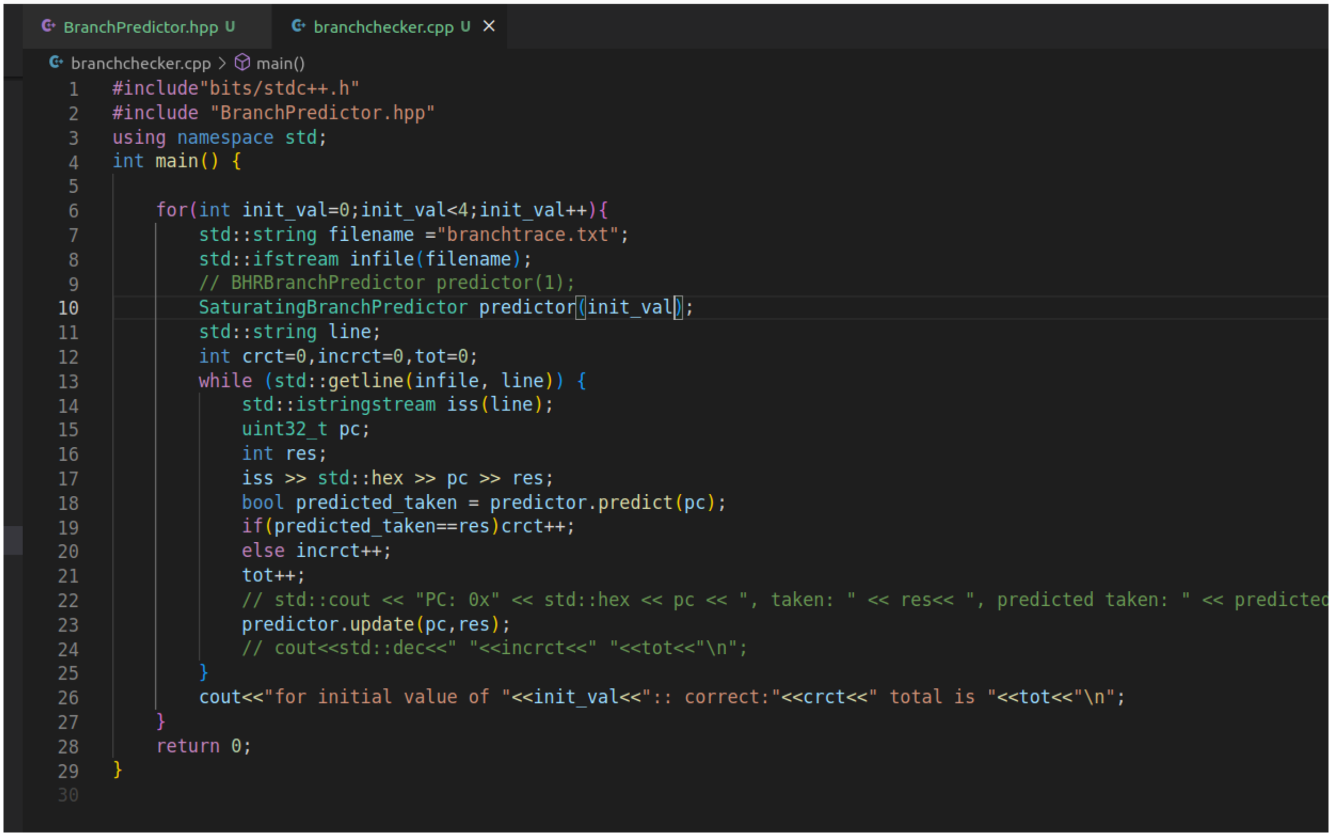
For 2bit predictor have initialised all the indexes with a starting value between 0 and 3. To predict for a given pc we first find its index by doing its bitwise and with table size minus 1.Then we check if the msb of that index is set we predict branch taken other wise not taken. Now for updating we similarly find its index,if the branch is taken in reality we need to increment the counter if it's less than 3 , and if branch is not taken we need to decrement the counter if it's greater than 0.



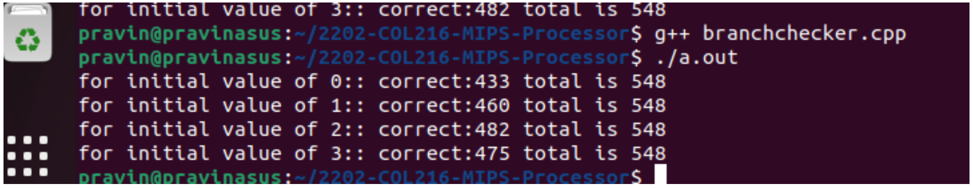
```
struct SaturatingBranchPredictor : public BranchPredictor {
    std::vector<std::bitset<2>> table;
    SaturatingBranchPredictor(int value) : table(1 << 14, value) {}

    bool predict(uint32_t pc) {
        uint32_t index = pc & (table.size()-1);
        return table[index].to_ulong()>=2;
        bool taken = table[index].test(1);
        return taken;
    }

    void update(uint32_t pc, bool taken) {
        uint32_t index = pc & (table.size()-1);
        if (taken && table[index].to_ulong() < 3)table[index]=bitset<2>(table[index].to_ulong() + 1);
        else if(!(taken) && (table[index].to_ulong()>0))table[index]=bitset<2>(table[index].to_ulong()-1);
    }
};
```

result for two bit saturating counter

For checking I have checked with all 4 starting values of the counter and we can see that accuracy is highest when the start counter is set to 2 for all the indexes which is weakly taken.



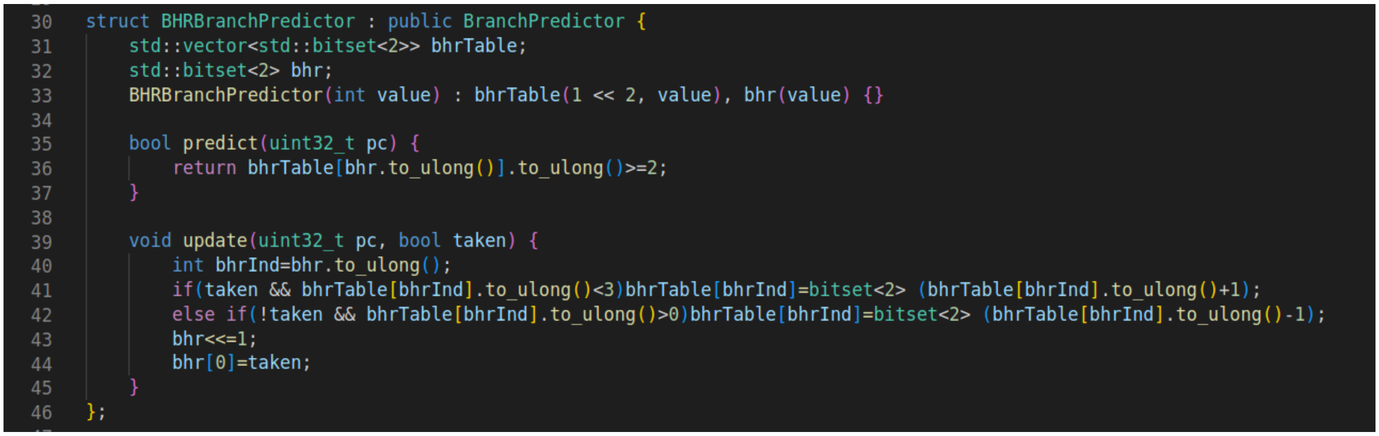
```
#include"bits/stdc++.h"
#include "BranchPredictor.hpp"
using namespace std;
int main() {

    for(int init_val=0;init_val<4;init_val++){
        std::string filename ="branchtrace.txt";
        std::ifstream infile(filename);
        // BHRBranchPredictor predictor(1);
        SaturatingBranchPredictor predictor(init_val);
        std::string line;
        int crct=0,incrct=0,tot=0;
        while (std::getline(infile, line)) {
            std::istringstream iss(line);
            uint32_t pc;
            int res;
            iss >> std::hex >> pc >> res;
            bool predicted_taken = predictor.predict(pc);
            if(predicted_taken==res)crct++;
            else incrct++;
            tot++;
            // std::cout << "PC: 0x" << std::hex << pc << ", taken: " << res<< ", predicted taken: " << predicted
            predictor.update(pc,res);
            // cout<<std::dec<<" "<<incrct<<" "<<tot<<"\n";
        }
        cout<<"for initial value of "<<init_val<<":: correct:"<<crct<<" total is "<<tot<<"\n";
    }
    return 0;
}
```



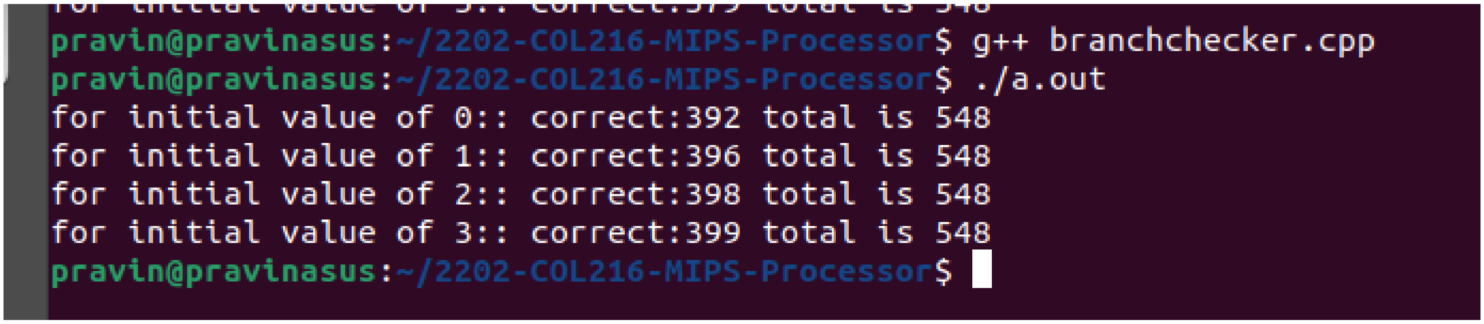
```
for initial value of 3:: correct:482 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:433 total is 548
for initial value of 1:: correct:460 total is 548
for initial value of 2:: correct:482 total is 548
for initial value of 3:: correct:475 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$
```

BHR Branch Predictor

Similar to the previous implementation we check after initialising all the indexes with a staring value between 0 and 3. To predict for a given pc, pc doesn't play any role here we first check the bhr and see what does bhrTable at that index suggest.If it's msb is 1 or it's value greater than equal to 2. We take the branch otherwise we don't take the branch.Now for updating we update the bhrtable index corresponding to the branch taken or not. And finally update the bhr value.



```
struct BHRBranchPredictor : public BranchPredictor {
    std::vector<std::bitset<2>> bhrTable;
    std::bitset<2> bhr;
    BHRBranchPredictor(int value) : bhrTable(1 << 2, value), bhr(value) {}

    bool predict(uint32_t pc) {
        return bhrTable[bhr.to_ulong()].to_ulong()>=2;
    }

    void update(uint32_t pc, bool taken) {
        int bhrInd=bhr.to_ulong();
        if(taken && bhrTable[bhrInd].to_ulong()<3)bhrTable[bhrInd]=bitset<2> (bhrTable[bhrInd].to_ulong()+1);
        else if(!taken && bhrTable[bhrInd].to_ulong()>0)bhrTable[bhrInd]=bitset<2> (bhrTable[bhrInd].to_ulong()-1);
        bhr<<=1;
        bhr[0]=taken;
    }
};
```



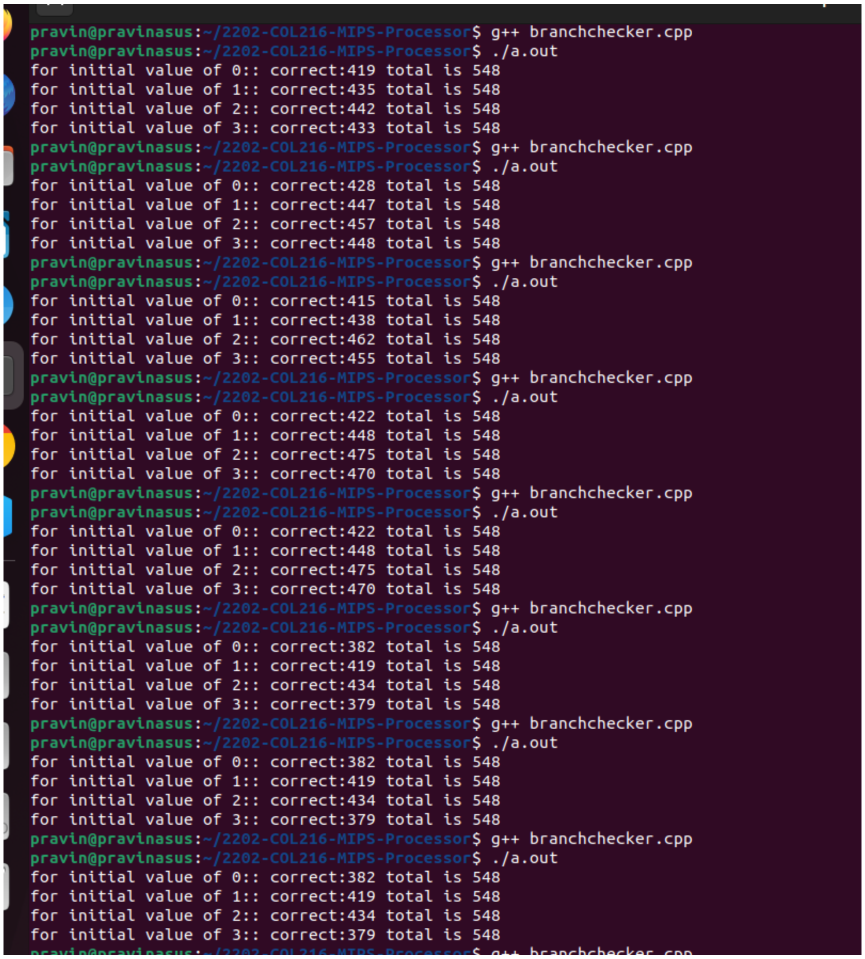
```
for initial value of 3:: correct:379 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:392 total is 548
for initial value of 1:: correct:396 total is 548
for initial value of 2:: correct:398 total is 548
for initial value of 3:: correct:399 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ 
```

Accuracy of BHR predictor

We run it on all 4 cases of starting value on the branchtrace.txt shared with us.And the result we obtained is as follows.We can observe that branch saturated register is slightly more accurate than branch history register in this case.

BHRB Saturated Predictor

We have tried to check for various linear and non linear combination of the above two implementation for this part but no implementation is better in accuralcy than the saturated predictor we implemented in the first part.



```
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:419 total is 548
for initial value of 1:: correct:435 total is 548
for initial value of 2:: correct:442 total is 548
for initial value of 3:: correct:433 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:428 total is 548
for initial value of 1:: correct:447 total is 548
for initial value of 2:: correct:457 total is 548
for initial value of 3:: correct:448 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:415 total is 548
for initial value of 1:: correct:438 total is 548
for initial value of 2:: correct:462 total is 548
for initial value of 3:: correct:455 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:422 total is 548
for initial value of 1:: correct:448 total is 548
for initial value of 2:: correct:475 total is 548
for initial value of 3:: correct:470 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:422 total is 548
for initial value of 1:: correct:448 total is 548
for initial value of 2:: correct:475 total is 548
for initial value of 3:: correct:470 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:382 total is 548
for initial value of 1:: correct:419 total is 548
for initial value of 2:: correct:434 total is 548
for initial value of 3:: correct:379 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:382 total is 548
for initial value of 1:: correct:419 total is 548
for initial value of 2:: correct:434 total is 548
for initial value of 3:: correct:379 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ ./a.out
for initial value of 0:: correct:382 total is 548
for initial value of 1:: correct:419 total is 548
for initial value of 2:: correct:434 total is 548
for initial value of 3:: correct:379 total is 548
pravin@pravinasus:~/2202-COL216-MIPS-Processor$ g++ branchchecker.cpp
```

Table for various implementation

accuracy for a starting value and 3 implementation			
Starting Value	Accuracy of Saturating Branch Predictor	Accuracy for BHR Predictor	Accuracy for Saturating BHR Predictor)
0	79	72	74
1	84	72	79
2	88	73	83
3	87	73	82

Distrubtion

We have contributed equally.