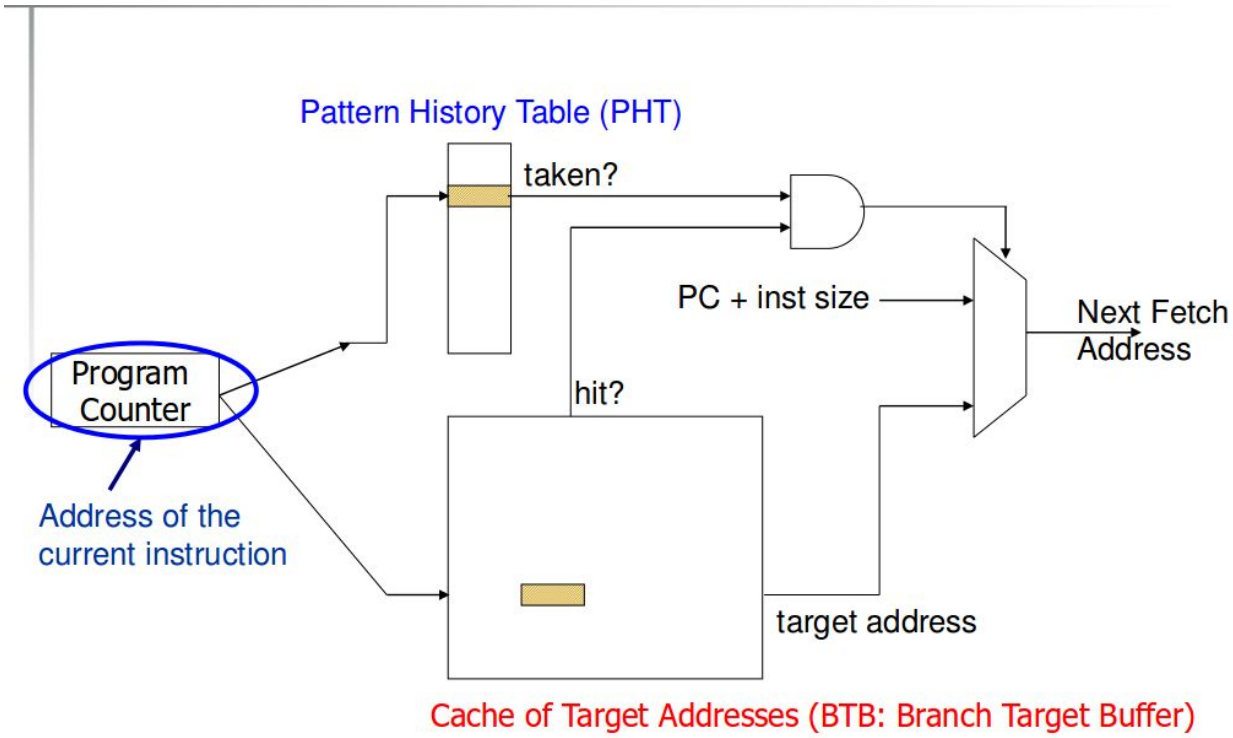
Assignment 2

Abhishek Nikam
800959985

1. A) Assume 1K (1024) entry for Pattern History Table (PHT) with 2-bit Counter per entry (predictor index size of 2 bits), implement and compare the mis-prediction of the following branch predictors.

One level branch Prediction.



- For One level branch predictor, we index into the pattern history table using only 10 bits of program counter as we have only 1024 PHT entries
- There are a total of 16 million instructions, we divide that number with the total of mispredictions to get the miss ratio.
- With a pattern history table of size 1024 and 2 bit saturating counters we get a miss ratio of 15.78
-
```
bhishek@abhishek-Inspiron-5559:~/Comp-arch/Homework2$ gcc Onelevel.c -o One
bhishek@abhishek-Inspiron-5559:~/Comp-arch/Homework2$ ./One
issratio is  15.787866abhishek@abhishek-Inspiron-5559:~/Comp-arch/Homework2$
```
- We use **(address>>1)%1024** for indexing into pht.
- Right shift is done to avoid aliasing.
- The effect of increasing the pattern history table size is shown below.
-
```
Missratio is  13.536570
```
- When we double our pht size we get a better miss ratio.

Assignment 2

Abhishek Nikam
800959985

C code

```c
#include <stdio.h>

long int prediction(unsigned long int address, unsigned char instr);

unsigned int pht[1024];
long int misprediction=0;


/* This function initializes pht to 0*/

void pht_init()
{
        int i;
        for(i=0;i<1024;i++)
        {
                pht[i]=0;
        }
}


int main()
{
        FILE *fp;
        int i, type;
        unsigned long int address;
        unsigned char instr;
        float branches=16416279;
        pht_init();

        fp = fopen("branch-trace-gcc.trace", "r"); // Opening the trace file.

        while (!feof(fp))                          //Keep on reading the file till Eof.
        {

        fscanf(fp, "%ld %c",  &address, &instr);
        misprediction=prediction(address,instr);
        }

        fclose(fp);

        printf("Missratio is  %f",(misprediction/branches)*100);


}
```

Assignment 2

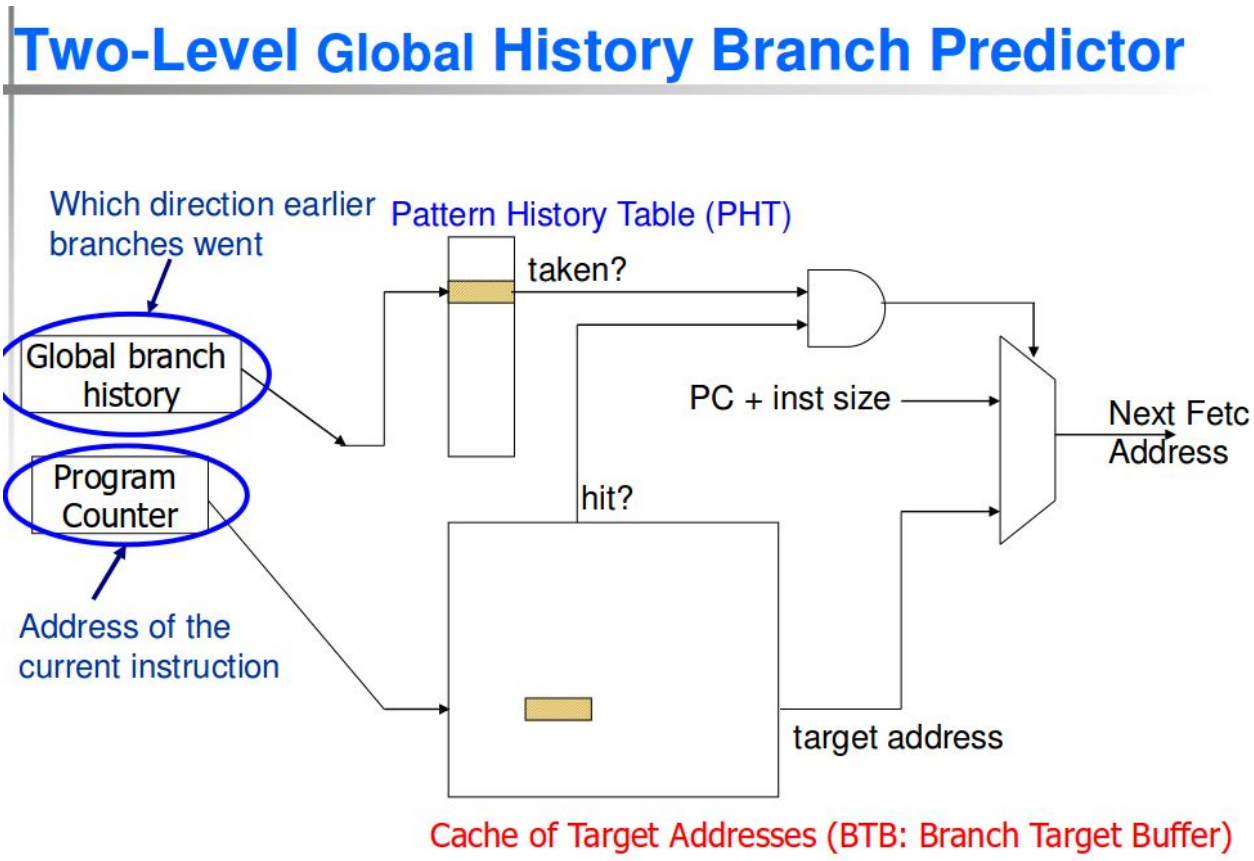
Abhishek Nikam
800959985

```
long int prediction(unsigned long int address, unsigned char instr)
        {

                unsigned long int address1;
                address1= (address>>1)%1024; // Right shift done to decrease aliasing


                if(instr=='T')
                {
                        if(pht[address1]==0 || pht[address1]==1)
                        {
                                misprediction++;
                        }

                        pht[address1]++;

                        if(pht[address1]>3)
                                {
                                        pht[address1]=3;
                                }

                }


                else if(instr=='N')
                {

                        if(pht[address1]==2 || pht[address1]==3)
                        {
                                misprediction++;
                        }

                        if(pht[address1]<1)
                        {
                                pht[address1]=0;
                        }
                        else
                        {
                                pht[address1]--;
                        }
                }

                return misprediction;
        }
```

Assignment 2

Abhishek Nikam
800959985

Two Level Global Branch Prediction



- In Two level Global History Branch Predictor we use a Global History Register of size 10 bits.
- In Two level Global History Branch Predictor we use Global Branch History to index into the pattern history table.
- The global history register will have the history of what the previous 10 branches did.
- If a branch was taken we would add 1 to the global history register, if it was not taken we will add 0 to the history register.

Assignment 2

Abhishek Nikam
800959985

- So now with pht size as 1024 and our global history register width as 10 bits the miss ratio we got is
- ```
  Missratio for pht size 1024 and GHR 10 bits is 22.567240
  ```
- The miss ratio is really bad compared to what we had got for One level predictor.
- This is because the size of pattern history table is too small and also because the global history register can only record the pattern of past 10 branches.
- We get much better results when we change the size of pattern history table to 4096 and the 12 bits for global history registers.

- ```
  Missratio for pht size 4096 and GHR 12 bits is 15.954377
  ```

Assignment 2

Abhishek Nikam
800959985

Code (PHT= 1024 and GHR as 10 bits)

```
#include <stdio.h>

long int prediction(unsigned long int address, unsigned char instr);
unsigned int pht[1024];
long int misprediction=0;
struct globalprediction
{
   unsigned int ghr:10;
} globalpredictor;

void pht_init()
{
        int i;
        for(i=0;i<1024;i++)
        {
                pht[i]=0;
        }
        globalpredictor.ghr=0;
}


int main()
{
        FILE *fp;
        int i, type;
        unsigned long int address;
        unsigned char instr;
        float branches=16416279;
        pht_init();
        //ghr_init();
        // open trace file for reading
        fp = fopen("branch-trace-gcc.trace", "r");


        while (!feof(fp))
        {

        fscanf(fp, "%ld %c",  &address, &instr);
        misprediction =prediction(address,instr);

        }

  fclose(fp);


  printf("Missratio for pht size 4096 and GHR 12 bits is %f\n",(misprediction/branches) *100);
}


long int prediction(unsigned long int address, unsigned char instr)
        {

                unsigned int branch;
```

Assignment 2

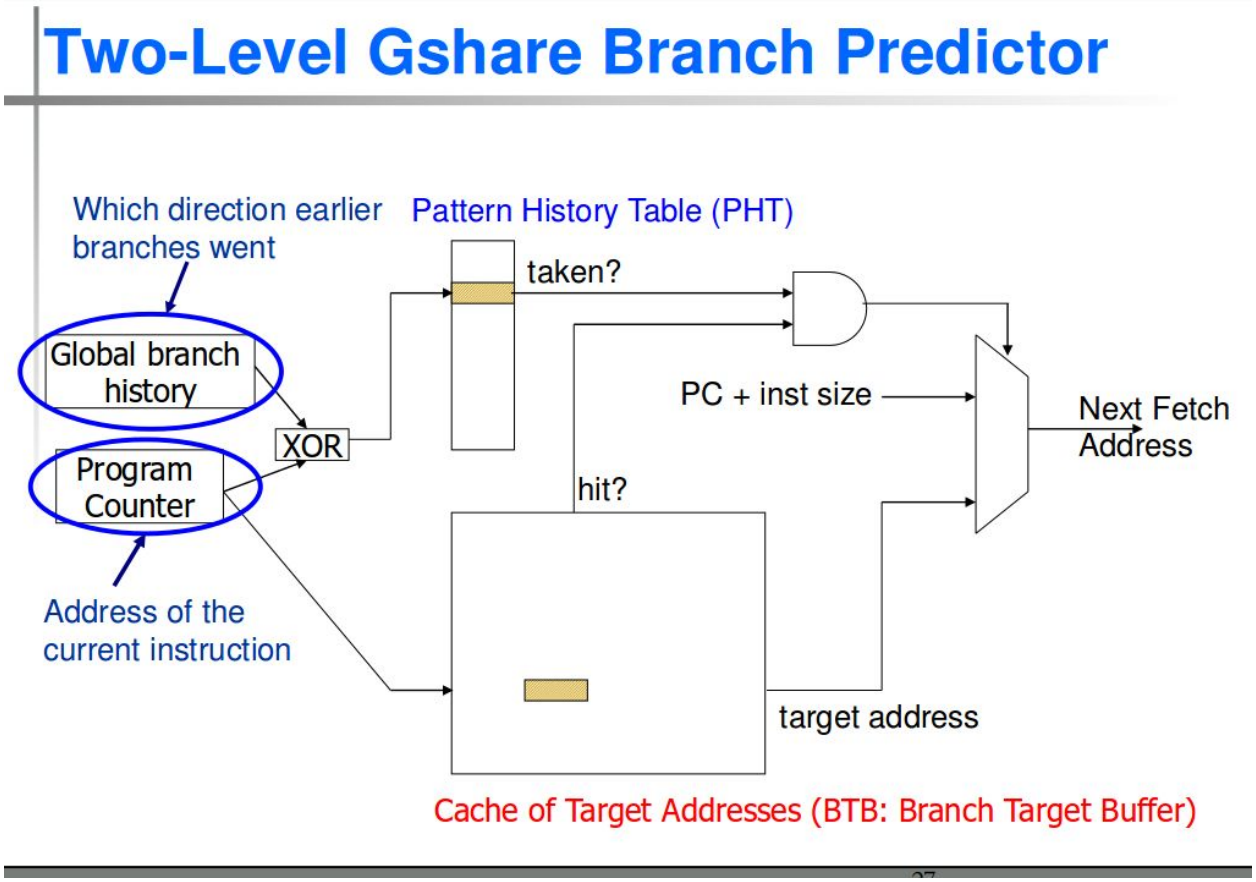
Abhishek Nikam
800959985

```
branch=pht[globalpredictor.ghr];
//printf("branch are %d", branch);


        if(instr=='T')
        {

                if(branch<524288)         //half value
                {

                        misprediction++;

                }

                pht[globalpredictor.ghr]++;
                if(pht[globalpredictor.ghr]>1048575)    // final value -1
                        {
                                pht[globalpredictor.ghr]=1048575;    // final value -1
                        }
                globalpredictor.ghr= globalpredictor.ghr<<1 | 1;

        }



        if(instr=='N')
        {
                if(branch>524287)     // half value-1
                {

                        misprediction++;

                }

                        if (pht[globalpredictor.ghr]<1)
                        {
                                pht[globalpredictor.ghr]=0;
                        }
                        else
                        {
                                pht[globalpredictor.ghr]--;
                        }

                        globalpredictor.ghr=globalpredictor.ghr<<1;


        }
```

Assignment 2

Abhishek Nikam
800959985

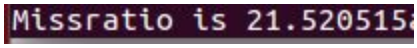
Two Level Gshare Branch Predictor

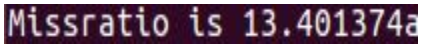


- In Gshare Branch Predictor, we Xor the Pc and Global Branch History to index into the pattern history.
- We do an Xor as an attempt to get more context about a particular Pc.
- This is a subtle way to avoid aliasing.
- We select any 10 bits of Pc and Xor it with Global Branch History as the size of pht is only 1024.

Assignment 2

Abhishek Nikam
800959985

- The advantage of using Gshare Branch predictor is
  - More context information
  - Better utilization of PHT
  - Increases access latency
- For Gshare branch predictor, with pht size as 1024 and global history register width as 10 bit we get a missratio of 21.52.
- 

- But when pht is 1024 and ghr has 12 bits we get a much better missratio.
- 

.

Assignment 2

Abhishek Nikam
800959985

C code

```c
#include <stdio.h>

long int prediction(unsigned long int address, unsigned char instr);
unsigned int pht[1024];
long int misprediction=0;

struct globalprediction
{
   unsigned int ghr:10;
} globalpredictor;



void pht_init()
{
        int i;
        for(i=0;i<1024;i++)
        {
                pht[i]=0;
        }
        globalpredictor.ghr=0;
}


int main()
{
        FILE *fp;
        int i, type;
        unsigned long int address;
        unsigned char instr;
        unsigned int masked_addr;
        float branches=16416279;
        pht_init();

        // open trace file for reading
        fp = fopen("branch-trace-gcc.trace", "r");



        while (!feof(fp))
        {

        fscanf(fp, "%ld %c",  &address, &instr);
        misprediction =prediction(address,instr);
        }

  fclose(fp);
  printf("Missratio is %f",(misprediction/branches) *100);
}


long int prediction(unsigned long int address, unsigned char instr)
        {
```

Assignment 2

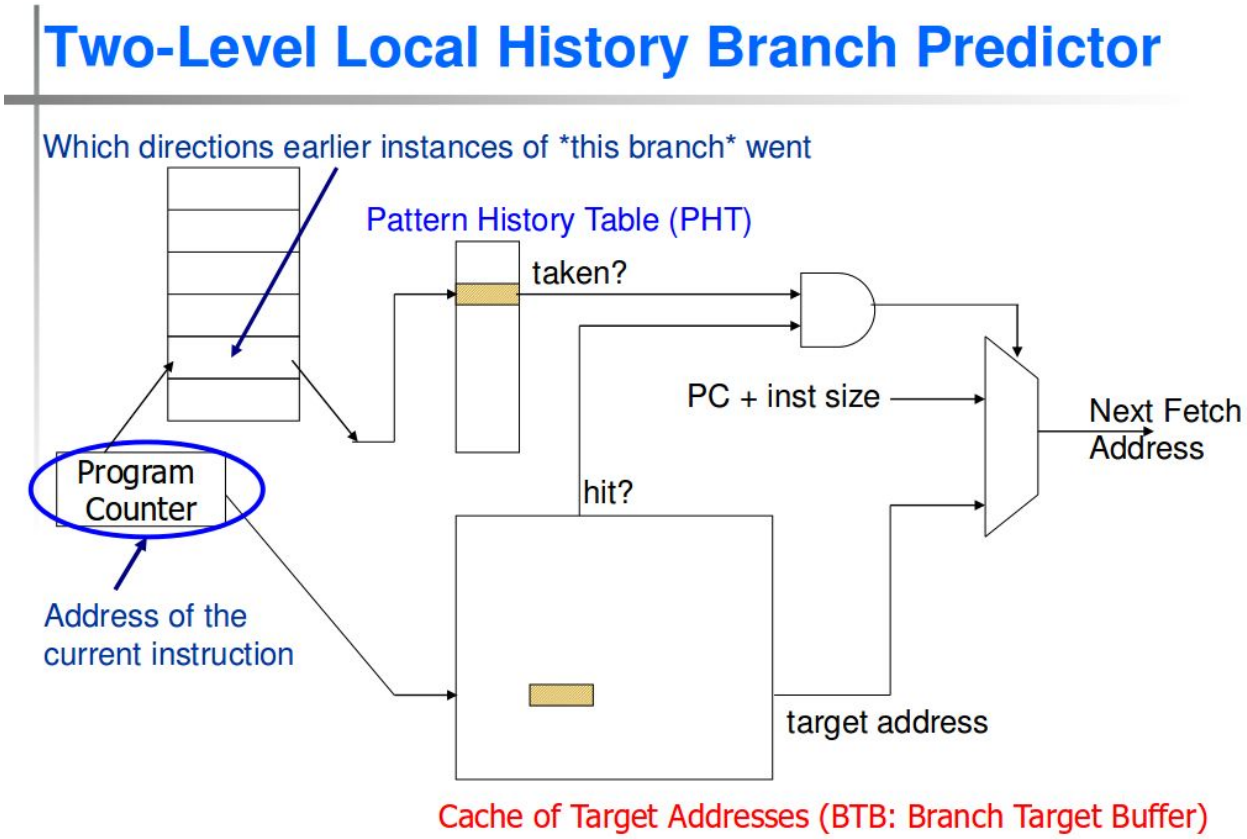
Abhishek Nikam
800959985

```
unsigned int branch;
branch=pht[globalpredictor.ghr];
//printf("branch are %d", branch);


        if(instr=='T')
        {

                if(branch<524288)         //half value
                {

                        misprediction++;

                }

                pht[globalpredictor.ghr]++;
                if(pht[globalpredictor.ghr]>1048575)    // final value -1
                        {
                                pht[globalpredictor.ghr]=1048575;    // final value -1
                        }
                globalpredictor.ghr= globalpredictor.ghr<<1 | 1;

        }



        if(instr=='N')
        {
                if(branch>524287)     // half value-1
                {

                        misprediction++;

                }

                        if (pht[globalpredictor.ghr]<1)
                        {
                                pht[globalpredictor.ghr]=0;
                        }
                        else
                        {
                                pht[globalpredictor.ghr]--;
                        }

                        globalpredictor.ghr=globalpredictor.ghr<<1;
```

Assignment 2

Abhishek Nikam
800959985

Two Level Local Branch Prediction



- In_Two level History Branch Predictor we use a local history table to keep track of what decision that branch took during last 10 times.
- We use that local history table value to index into PHT.
- This predictor just keeps track of what that branch did previously nowhere is it concerned with the global history.
- The size of our local history table is 128 and each field inside it has 10 bits.
- As the size of our local history table is 128, we use 7 bits of our Pc to index into it.
- With the given pht size and local history table size we get a missratio of **23.113.**
- The missratio can be significantly improved if we increase the size of our local history table and pattern history table.

Assignment 2

Abhishek Nikam
800959985

- As we increase the size of local history table, the width of it we can keep more information about the decision taken by branches.
- I changed the pht size to 16384 and local history table size to 2048 to observe the changes in missratio and the results were astonishing.
- The missratio observed was only **8.9.**

C code

```
#include <stdio.h>
long int prediction(unsigned long int address, unsigned char instr);
unsigned int pht[16384];
long int misprediction=0;


struct localprediction
{
   unsigned int lhr:14;
} localpredictor[2048];



void pht_init()
{
        int i;
        for(i=0;i<16384;i++)
        {
                pht[i]=0;
        }

}

void local_init()
{
        int i;
        for(i=0;i<2048;i++)
        {
                localpredictor[i].lhr=0;
        }
}

int main()
{
        FILE *fp;
        int i, type;
        unsigned long int address;
        unsigned char instr;
        unsigned int masked_addr;
        float branches=16416279;
        pht_init();
        local_init();

        fp = fopen("branch-trace-gcc.trace", "r");
        while (!feof(fp))
        {
        fscanf(fp, "%ld %c",  &address, &instr);
        misprediction =prediction(address,instr);
        }

  fclose(fp);
  printf("Missratio is %f",(misprediction/branches) *100);
}
```

Assignment 2

Abhishek Nikam
800959985

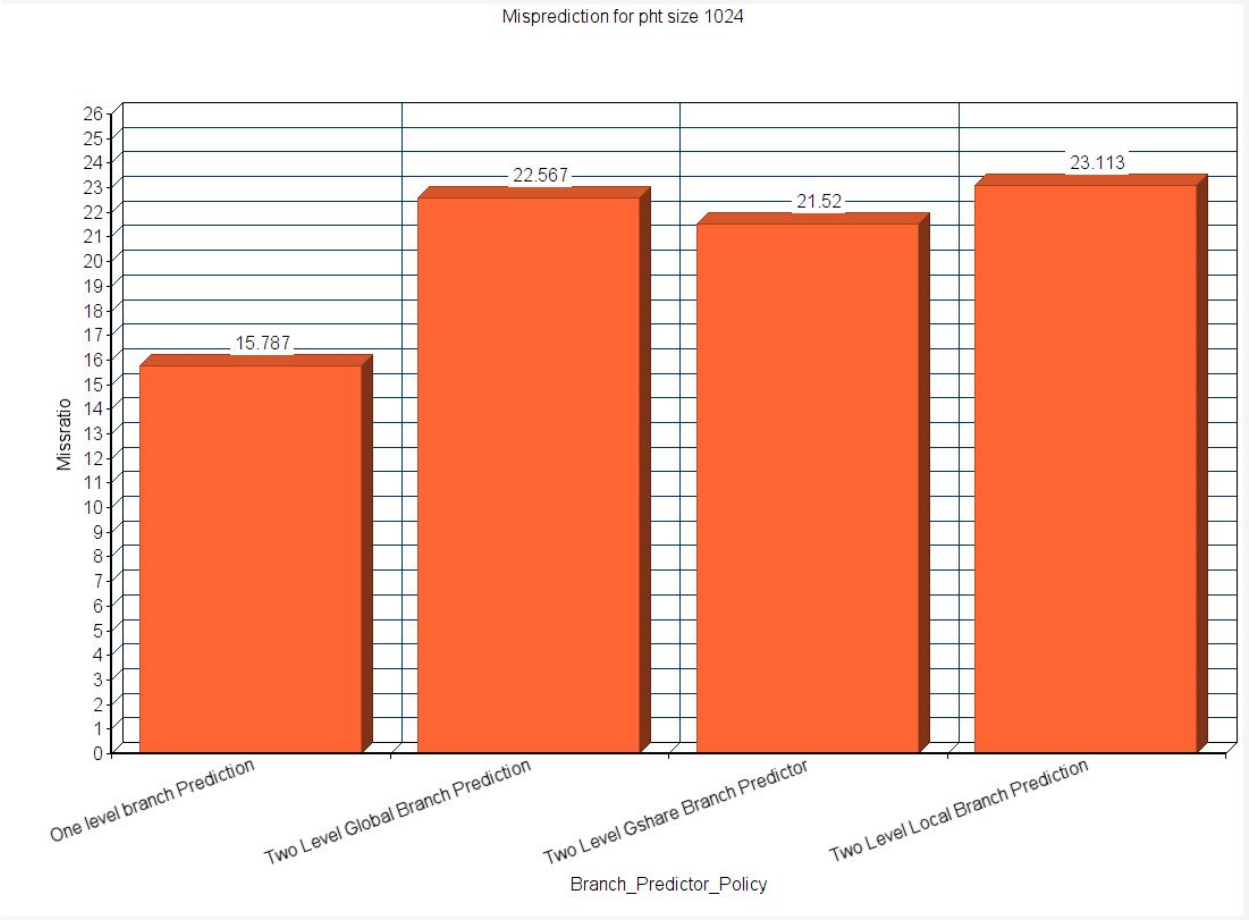
```
long int prediction(unsigned long int address, unsigned char instr)
        {
                unsigned  long int branch;
                unsigned long int pc= address % 2048;
                branch= localpredictor[pc].lhr;

                        if(instr=='T')
                        {

                                if(pht[branch]==0 || pht[branch]==1)
                                {

                                        misprediction++;

                                }


                                pht[branch]++;
                                if(pht[branch]>3)
                                        {
                                                pht[branch]=3;
                                        }
                                localpredictor[pc].lhr= localpredictor[pc].lhr<<1 | 1;

                        }



                        else if(instr=='N')
                        {
                                if(pht[branch]==2 || pht[branch]==3)
                                {

                                        misprediction++;

                                }


                                        if(pht[branch]<1)
                                        {
                                                pht[branch]=0;
                                        }
                                        else
                                        {
                                                pht[branch]--;
                                        }


                                        localpredictor[pc].lhr= localpredictor[pc].lhr <<1;

                        }
```

Assignment 2

Abhishek Nikam
800959985

Results

MissRatio for all predictors using pht size as 1024.



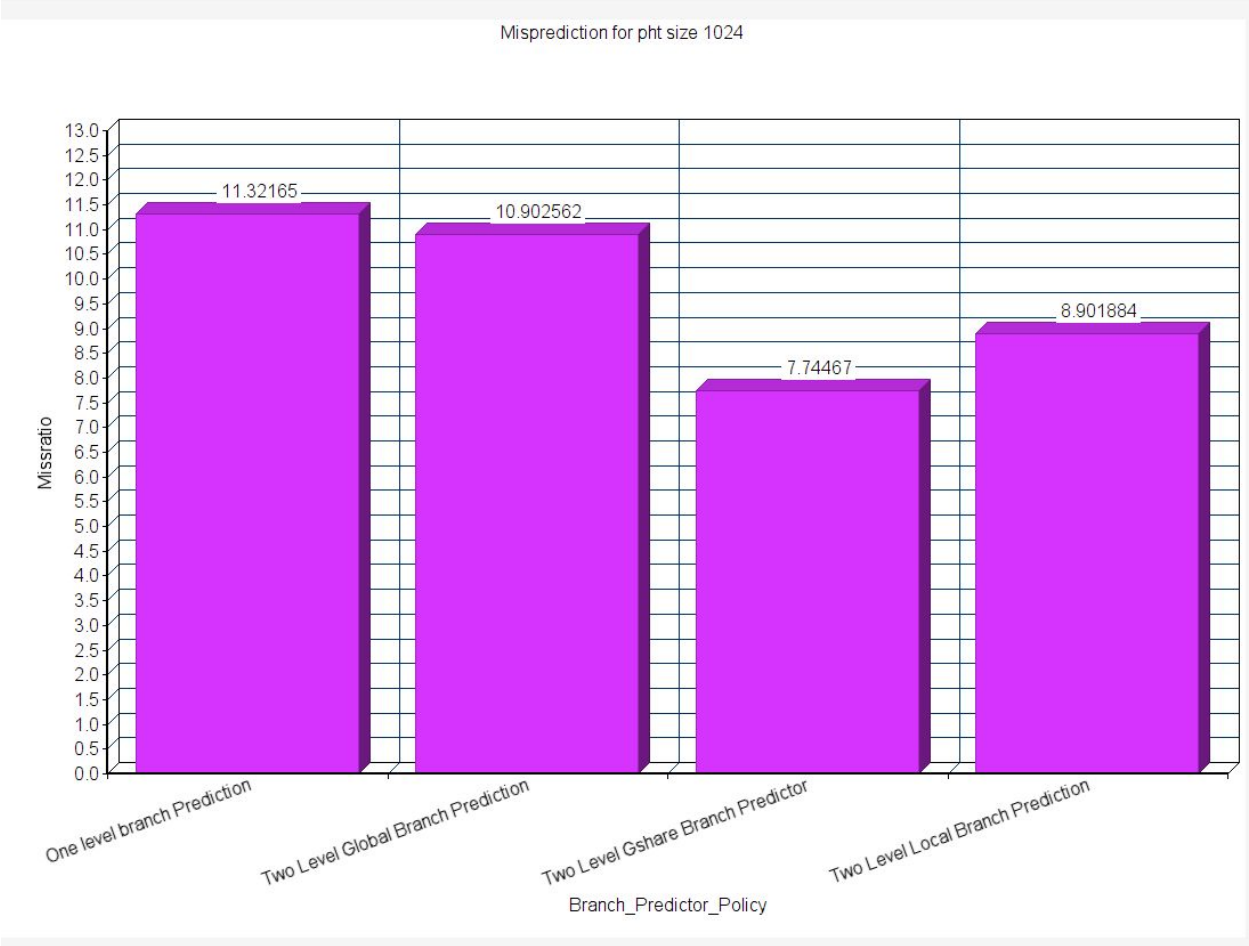
- From the results plotted above we can observe that **One level prediction** gives us the best result followed by Gshare prediction
- The **Two level Global Branch predictor** gives us not so good results because of the limited pht size and the width of global history registers.
- By increasing the pht size we can decrease the aliasing effect while by increasing the bit width of global history register we can keep more information about the decision taken by neighbouring branches which may yield better missratio.
- In **Two level Gshare Branch Predictor** we get a slight improvement in results as we are trying to decrease the aliasing effect by Xoring Ghr with Pc.
- But the size of pht and bitwidth of ghr seem to bring down the performance in it.
- **Two level Local Branch Predictor** technically gives us the highest missratio.

Assignment 2

Abhishek Nikam
800959985

- As we have only 128 entries in our Local history table and the size of each entry is only 10 bits we get such bad results.

Missratio for all predictors using pht size as 16384 and local history table size as 2048



Assignment 2

Abhishek Nikam
800959985

- As we increase the pht size to 16384, we observe that the performance for all branch predictors significantly improve.
- As Ghr is used to index into pht table we must also change ghr bitwidth from 10 bits to 14 bits.
- From the above plot we can observe that, **Two level Gshare Branch Predictor** gives us the best missratio.
- **Two level Local Branch predictor's** missratio also shows significant improvement.
- So, we can conclude that as we increase the size of pht the aliasing effect decreases.
- As we increase the bitwidth of global history register and local history register we can keep more history about that branch or neighbouring ones, so we observe the positive effect on missratio.

Assignment 2

Abhishek Nikam
800959985

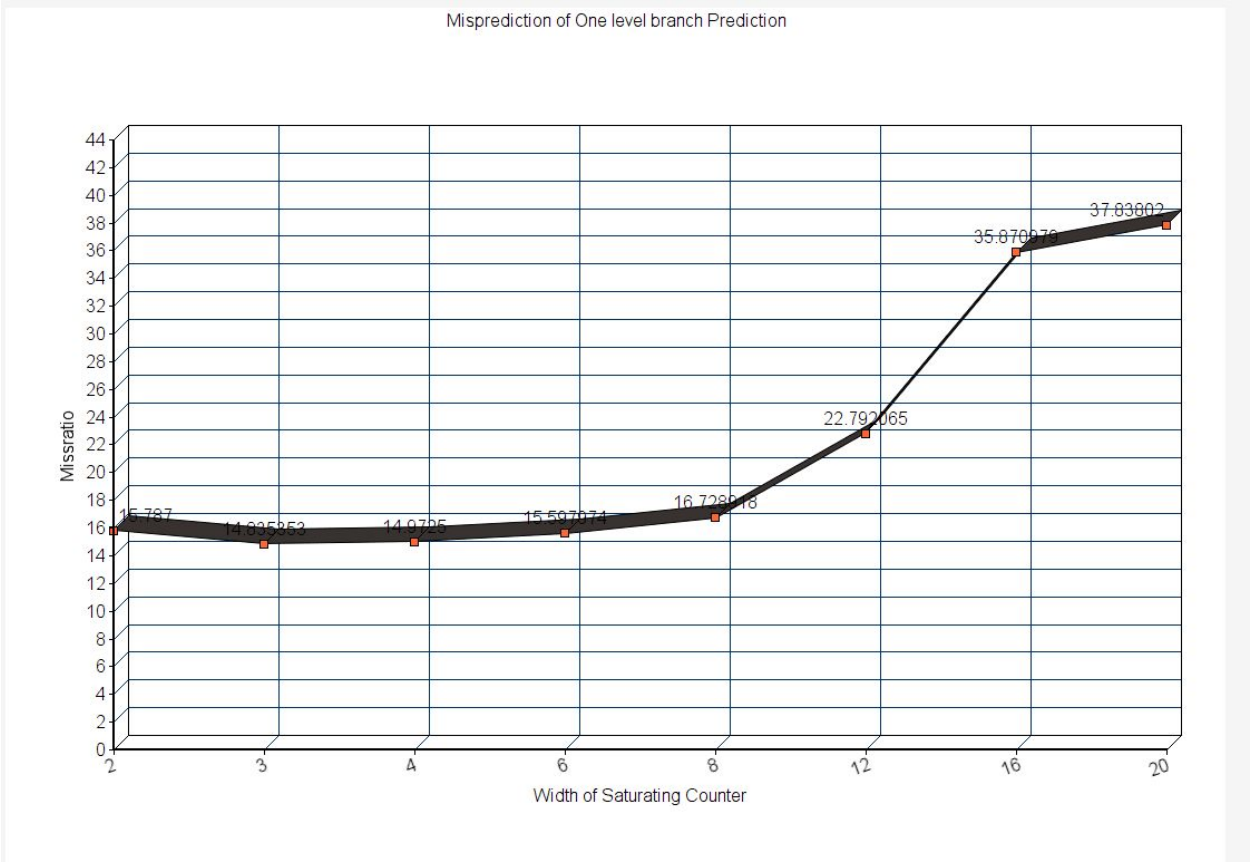
1.B) Repeat the Problem 1.A, this time with variable entry size (n-bit Counter) with size of 2 bits, 3 bits, 4 bits, 5 bits, ... 20 bits. Generate a dedicated line plot of the data using MS Excel or some other graphing program for each branch predictor. On the y-axis, plot "percentage of branches mis-predicted". On the x-axis plot the the predictor size (basically, the width of n-bit counter). Draw a separate plot for each branch predictor. Analyze the data and argue the effect of
increasing counter bits on branch predictor performance. In your view how many bits seems sufficiently enough for branch predictor (50 pts)?

- So, In this problem we will change the width of our saturating counter and plot the effect of it,

One level branch Predictor

Assignment 2

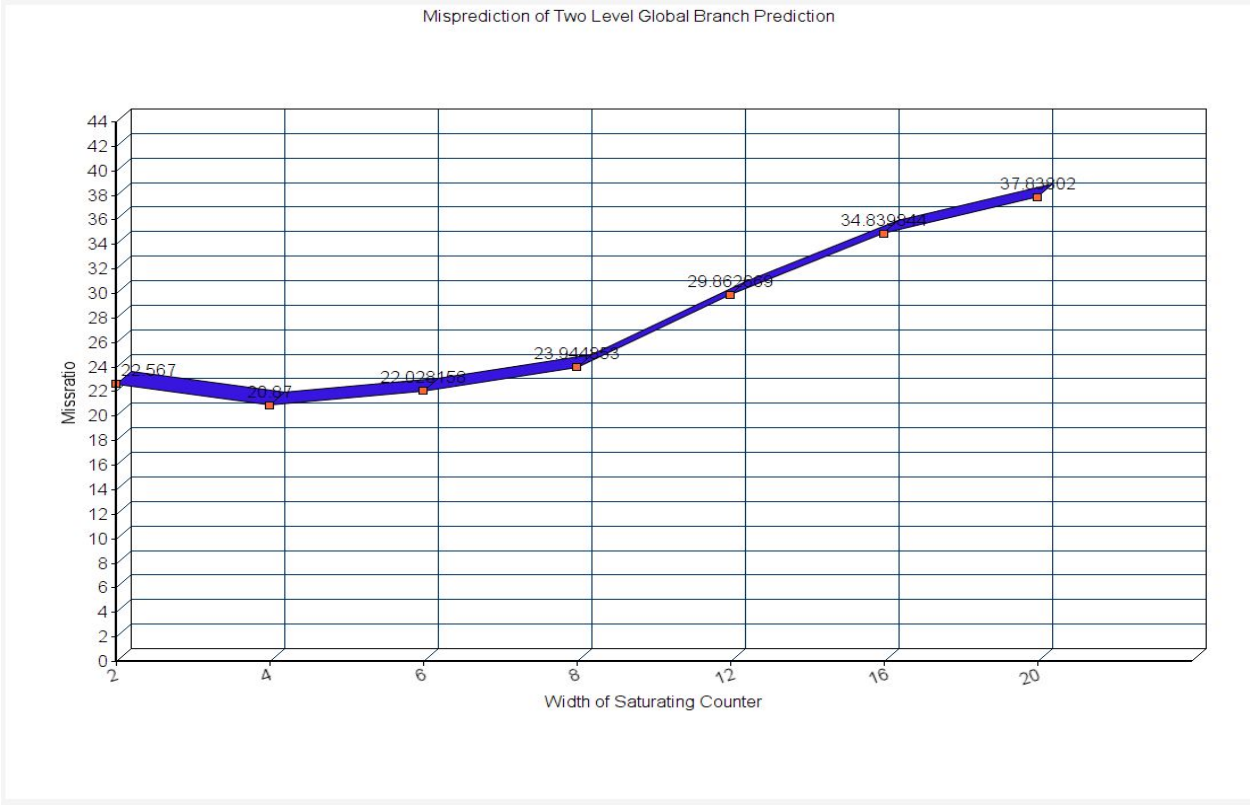
Abhishek Nikam
800959985



Two Level Global Branch Prediction

Assignment 2

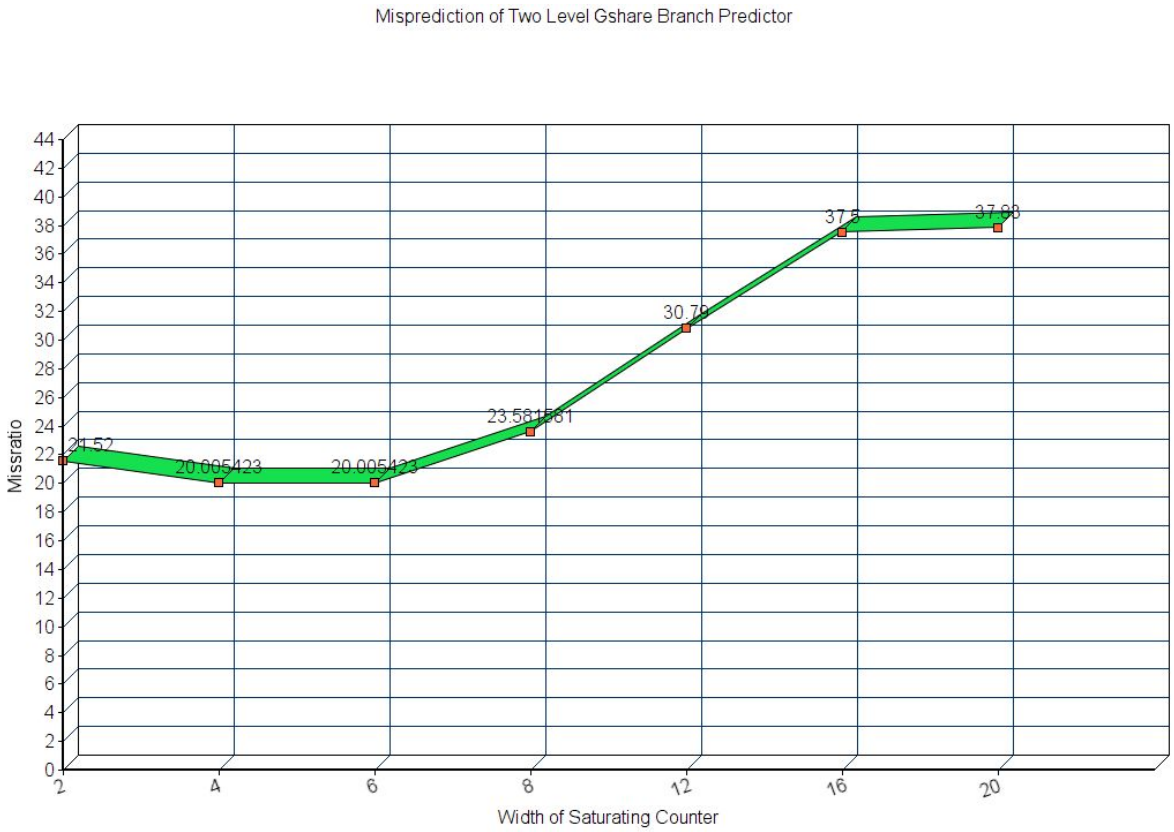
Abhishek Nikam
800959985



Two Level Gshare Branch Prediction

Assignment 2

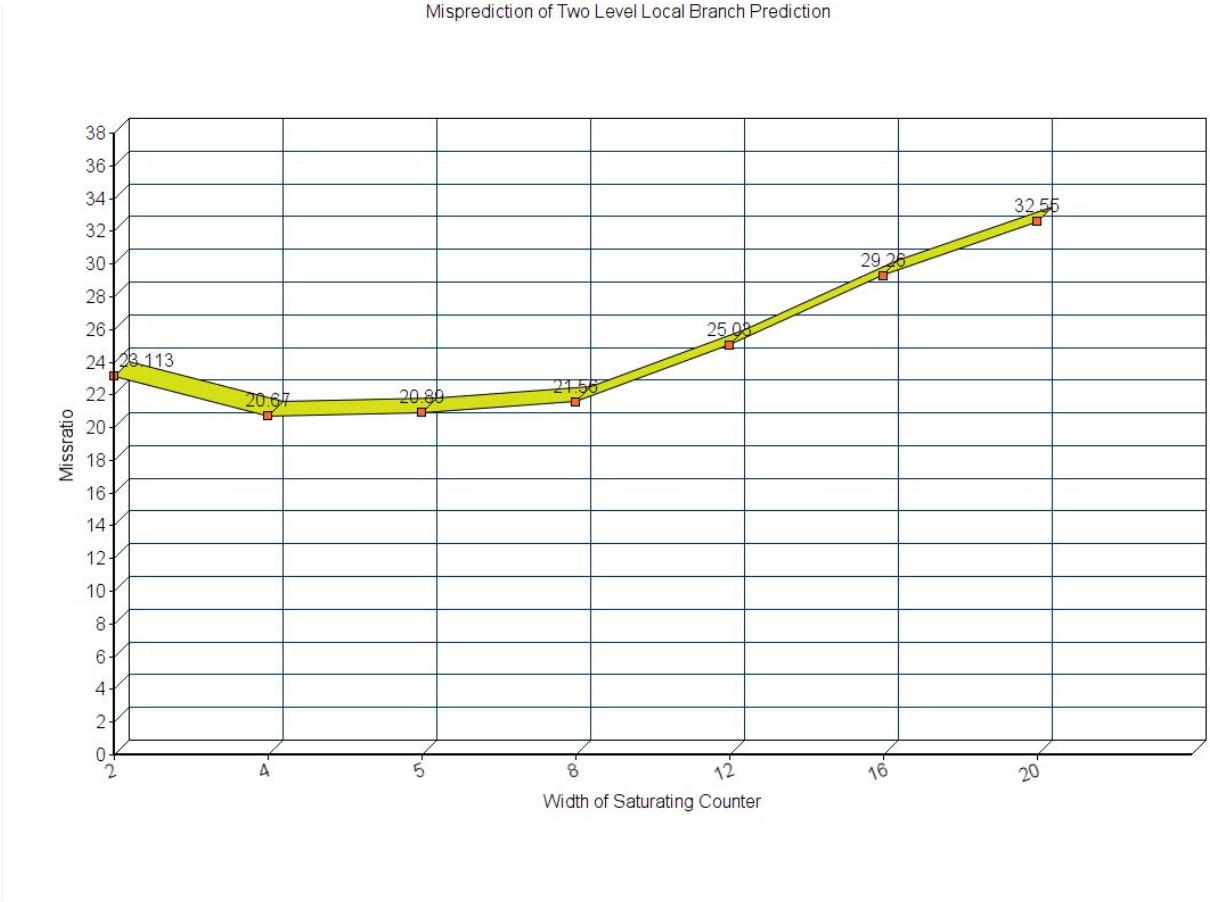
Abhishek Nikam
800959985



Two Level Local Branch Predictor

Assignment 2

Abhishek Nikam
800959985



- For this part of the question we change the bit width of n bit counter and observe the effect of that on the missratio.
- For a 2 bit saturating counter we have 4 states
    - 00 -> Strongly Not Taken
    - 01- -> Weakly Not Taken
    - 10 -> Weakly Taken
    - 11 -> Strongly Taken

- This means that for a transition from Strongly Not taken to Taken the branch must be taken atleast twice so that our counter value would be 01.
- For every time a branch is taken we add 1 to the counter and for every time a branch is not taken we decrement one from the counter.
- So, consider we have a 4 bit saturating counter, for a transition from strongly not taken to taken the branch must be taken at least 8 times.
- Above, I have plotted the graph of missratio when we use different sizes of saturating counter.

Assignment 2

Abhishek Nikam
800959985

- From the graph it is clearly visible that, we get the lowest missration when the width of saturating counter is 4.
- The miss ratio starts increasing as we keep on increasing the width of saturating counter because it requires a lot of mispredictions for transition from one state to another.
- So, according to me **4 bits** are enough for saturating counters.

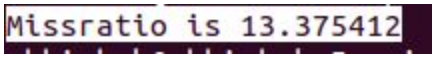
Assignment 2

Abhishek Nikam
800959985

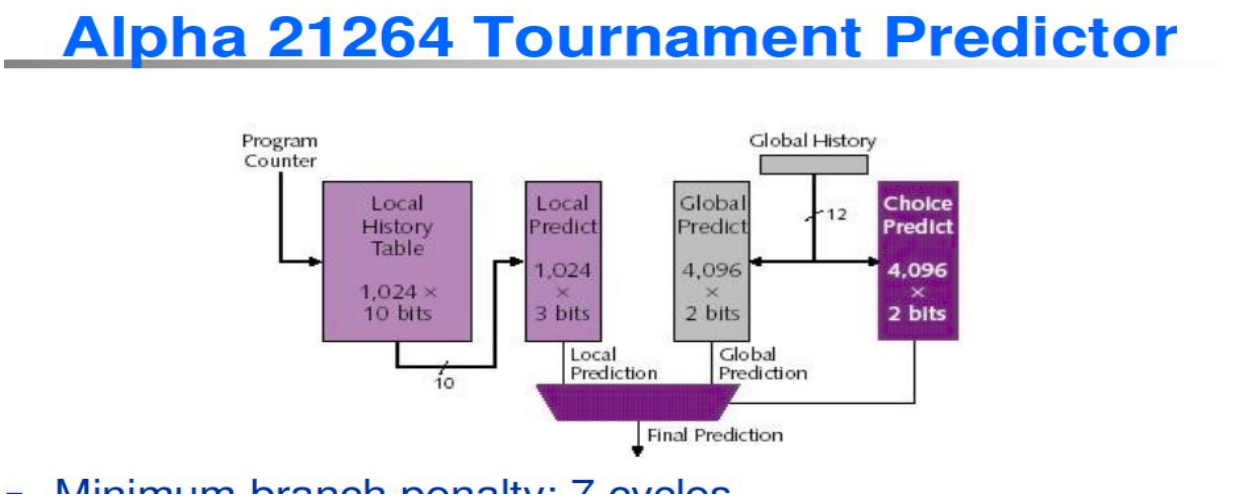
Extra Credit

Tournament Predictor.

Implement a branch predictor that combine the benefits of Gshare and Two Level Local Branch Predictor. Plot your result and compare it with previous branch predictor. Make sure to argue and
analyze you achieved results.

- I have implemented a tournament predictor for this part of question which combines the benefit of Gshare and Two Level Local Branch Predictor to give us better results.
- I have a 4K 2-bit counters to choose from among a global predictor and a local predictor.
- Global predictor also has 1K entries and is indexed by the history of the last 10 branches each entry in the global predictor is a standard 2-bit predictor.
- Local history table consisting of 1k 10-bit entries; each 10-bit entry corresponds to the most recent 10 branch outcomes for the entry. 10-bit history allows patterns 10 branches to be discovered and predicted.
- The performance achieved was better than all the predictors.
- 

- A tournament predictors reference diagram is given below



- The code I have written is **not** for Alpha 21264 Predictor.

Assignment 2

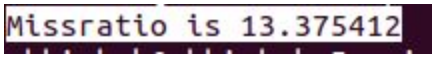
Abhishek Nikam
800959985

- Basically In my code, Initially I am biased towards using Local Predictor, as the Local Branch Predictor starts mispredicting, I switch on to Gshare Predictor.
- I use a saturating counter to switch between Branch Predictors.
- I am attaching a screenshot of the main part of the code where the Tournament predictor decides which Branch Predictor to use.

```
long int prediction(unsigned long int address, unsigned char instr)
{

        unsigned long int address1= address%4096; //Tournament table size is 4096.


        if ( tournament[address1]<2)             // We use a 2 bit saturating counter
        {
                misses= local(address,instr);
                if(flag==1)
                {

                        tournament[address1]++;
                        flag=0;
                }
                else
                {
                        if(tournament[address1]<1)
                        {
                                tournament[address1]=0;

                        }
                        else
                        {
                                tournament[address1]--;
                        }

                 }
        }


        if(tournament[address1]>1)
        {
                misses1= gshareprediction(address,instr);
                if(flag1==1)
                {
                        tournament[address1]--;
                        flag1=0;
                }

                else
                {

                        tournament[address1]++;
                        if(tournament[address1]>3)
                        {
                                tournament[address1]=3;
                        }

                }

        }
```

Assignment 2

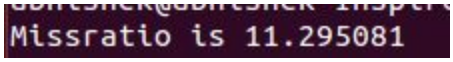
Abhishek Nikam
800959985

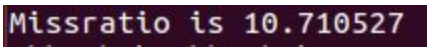
- The missratio when Tournament table size is 4096 and bitwidth of saturating counter is 2 and local history table size is 4096 is
- 

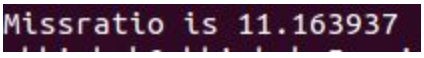
- Now, when I increase the bitwidth of my saturating counter to 3 I get better results,as it requires more transition to switch branch predictors.

Results

- For this tournament predictor we will observe the results by changing different configurations.
- We will keep Pattern history Table size as 1024.
- When our **local history table is fixed at size 1024** and the **global history register bit width is also 10 bits** and we have **4K 2-bit counters** to choose from among a global predictor and a local predictor.
- We get a 

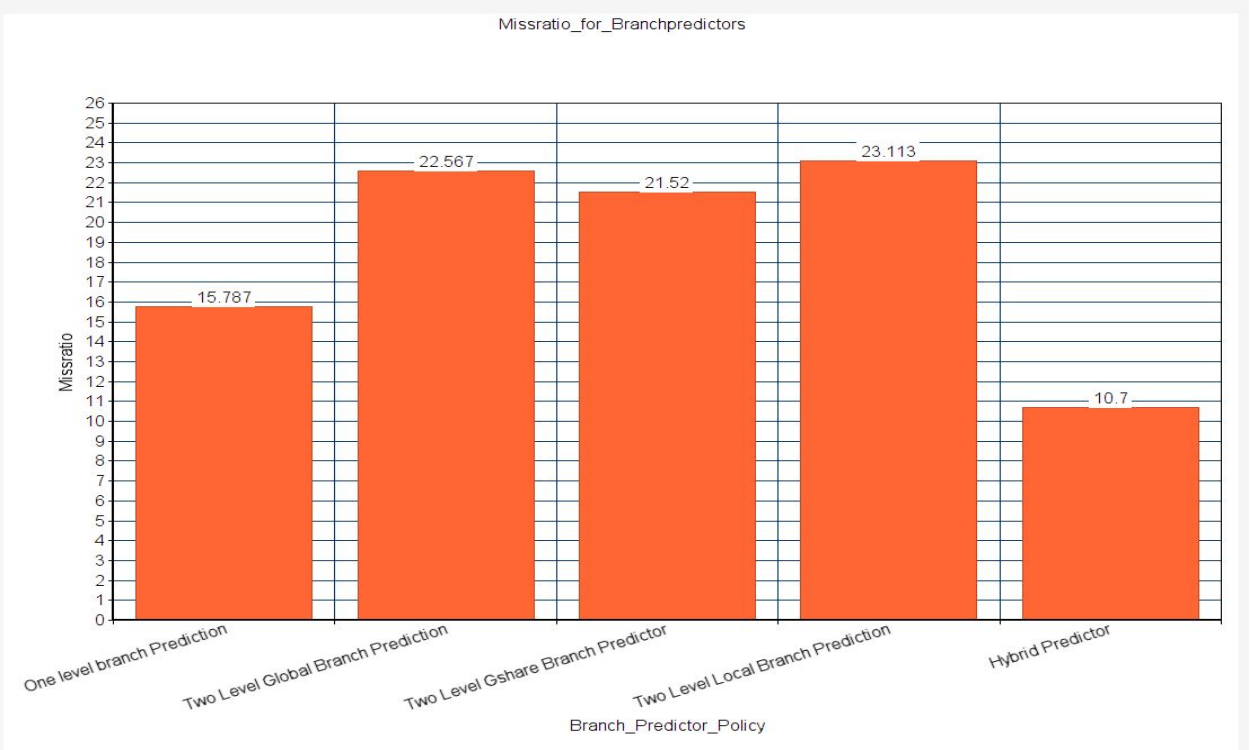
- If we increase the bitwidth of the counter which is used to choose from among a global predictor and a local predictor to 3 we get.
- 

- For a bitwidth of 4 we get 

.
- For a bitwidth of 5 we get 

Assignment 2

Abhishek Nikam
800959985



- From the above graph we can say that our Hybrid Predictor gives us the best missratio.