**To determine the processing efficiency (in number of bits per cycle) of**

**the “non-restoring” division technique with shift-over-0/1.**

**Program listing:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int num[33],i,j,h,n,d;

float eff;

float s,iter,avg,aors,avgaors;

int den[16],deninv[16],carry[17];

int quo[17];

int rem[17];

//Initializing Num and Den and Temporary Reminder for(n=16;n<=32;n=n+2)

for(n=16;n<=32;n=n+2)

{

d=n/2;

s=0;

aors=0;

for(iter=1;iter<=1000;iter++)

{

for(i=0;i<=(n-1);i++)

{

num[i]=rand()%2;

//printf(" Num[%d]=%d",i,num[i]);

}

num[n+1]=0;

//printf("\n Initial Reminder:\n"); for(i=0;i<=(d-1);i++)

for(i=0;i<=(d-1);i++)

{

rem[i]=num[i];

//printf(" Rem[%d]=%d",i,rem[i]);

}

den[0]=0;den[1]=1;deninv[0]=1;deninv[1]=0;

for(i=2;i<=(d-1);i++)

{

den[i]=rand()%2;

deninv[i]=1-den[i];

}

//9 Cycles at most for Sub n Shift

for(int cycle=0;cycle<=d;cycle++)

{

//printf("\n Cycle %d Reminder:\n",cycle);

for(i=0;i<=0;i++)//Because Den is 8 bits

{

if(rem[i]==0 && rem[i+1]==0)//Shift over

{

quo[cycle]=0;

s=s+1;

for(j=0;j<=(n-1);j++)

{

num[j]=num[j+1];

}

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

//printf(" Rem[%d]=%d",h,rem[h]);

}

}

else if(rem[i]==1 && rem[i+1]==1)//Shift over

{

quo[cycle]=1;

s=s+1;

for(j=0;j<=(n-1);j++)

{

num[j]=num[j+1];

}

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

//printf(" Rem[%d]=%d",h,rem[h]);

}

}

else if(rem[i]==0 && rem[i+1]==1)

{

aors=aors+1;

carry[d]=0;

for(j=(d-1);j>=0;j--)

{

carry[j]=(rem[j]&deninv[j])|(rem[j]&carry[j+1])|(deninv[j]&carry[j+1]);

rem[j]=rem[j]^deninv[j]^carry[j+1];

}

if (rem[0]<0)

quo[cycle]=0;

else if(rem[0]>=0)

quo[cycle]=1;

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

//printf(" Rem[%d]=%d",h,rem[h]);

}

}

else if(rem[i]==1 && rem[i+1]==0)

{

aors=aors+1;

carry[d]=0; for(j=(d-1);j>=0;j--)

{

carry[j]=(rem[j]&den[j])|(rem[j]&carry[j+1])|(den[j]&carry[j+1]);

rem[j]=rem[j]^den[j]^carry[j+1];

}

if (rem[0]<0)

{

quo[cycle]=0;

}

else if(rem[0]>=0)

{

quo[cycle]=1;

}

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

//printf(" Rem[%d]=%d",h,rem[h]);

}

}

}

}

}//For Iteration

avg=s/1000;

avgaors=aors/1000;

//printf("\nNo. of Shift Over cycles for n=%d : %f",n,avg);

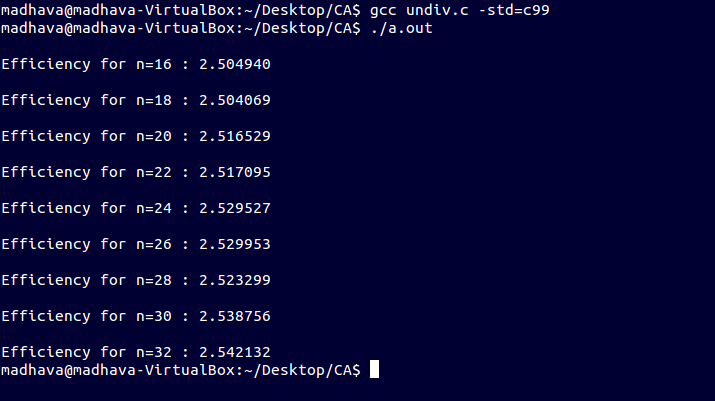
eff=(d+1.0)/avgaors;

printf("\nEfficiency for n=%d : %f\n",n,eff);

}

}

**Output:**

****

**Efficiency plot:**

**2. 16-bit division:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i,j,h,n,d;

float eff;

float s,iter,avg,aors,avgaors;

int deninv[16],carry[17];

int quo[17];

int rem[17];

//Initializing Num and Den and Temporary Reminder

for(n=32;n<=32;n=n+2)

{

d=n/2;

s=0;

aors=0;

for(iter=1;iter<=1;iter++)

{

int num[33]={0,0,0,0,1,0,0,1,1,0,1,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0};

printf(" Num[%d]=%d",i,num[i]);

printf("\n Initial Reminder:\n");

for(i=0;i<=(d-1);i++)

{

rem[i]=num[i];

printf(" Rem[%d]=%d\t",i,rem[i]);

}

int den[16]={0,1,1,1,1,0,1,1,0,0,1,0,1,0,1,0};

for(i=0;i<=(d-1);i++)

{

deninv[i]=1-den[i];

}

//9 Cycles at most for Sub n Shift

for(int cycle=0;cycle<=d;cycle++)

{

printf("\n\n Cycle %d Reminder:\n\n",cycle);

for(i=0;i<=0;i++)//Because Den is 8 bits

{

if(rem[i]==0 && rem[i+1]==0)//Shift over

{

quo[cycle]=0;

s=s+1;

for(j=0;j<=(n-1);j++)

{

num[j]=num[j+1];

}

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

printf(" Rem[%d]=%d\t",h,rem[h]);

}

}

else if(rem[i]==1 && rem[i+1]==1)//Shift over

{

quo[cycle]=1;

s=s+1;

for(j=0;j<=(n-1);j++)

{

num[j]=num[j+1];

}

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

printf(" Rem[%d]=%d\t",h,rem[h]);

}

}

else if(rem[i]==1 && rem[i+1]==0)

{

aors=aors+1;

carry[d]=0;

for(j=(d-1);j>=0;j--)

{

carry[j]=(rem[j]&den[j])|(rem[j]&carry[j+1])|(den[j]&carry[j+1]);

rem[j]=rem[j]^den[j]^carry[j+1];

}

if (rem[0]<0)

quo[cycle]=0;

else if(rem[0]>=0)

quo[cycle]=1;

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

printf("Rem[%d]=%d\t",h,rem[h]);

}

}

if(rem[i]==0 && rem[i+1]==1)

{

aors=aors+1;

carry[d]=0;

for(j=(d-1);j>=0;j--)

{

carry[j]=(rem[j]&deninv[j])|(rem[j]&carry[j+1])|(deninv[j]&carry[j+1]);

rem[j]=rem[j]^deninv[j]^carry[j+1];

}

if (rem[0]<0)

quo[cycle]=0;

else if(rem[0]>=0)

quo[cycle]=1;

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

printf(" Rem[%d]=%d\t",h,rem[h]);

}

}

else if(rem[i]==1 && rem[i+1]==0)

{

aors=aors+1;

carry[d]=0;

for(j=(d-1);j>=0;j--)

{

carry[j]=(rem[j]&den[j])|(rem[j]&carry[j+1])|(den[j]&carry[j+1]);

rem[j]=rem[j]^den[j]^carry[j+1];

}

if (rem[0]<0)

quo[cycle]=0;

else if(rem[0]>=0)

quo[cycle]=1;

rem[d]=num[d];

for(h=0;h<=(d-1);h++)

{

rem[h]=rem[h+1];

printf(" Rem[%d]=%d\t",h,rem[h]);

}

}

}

}

}//For Iteration

avg=s/1;

avgaors=aors/1;

printf("\nNo. of Shift Over cycles for n=%d : %f",n,avg);

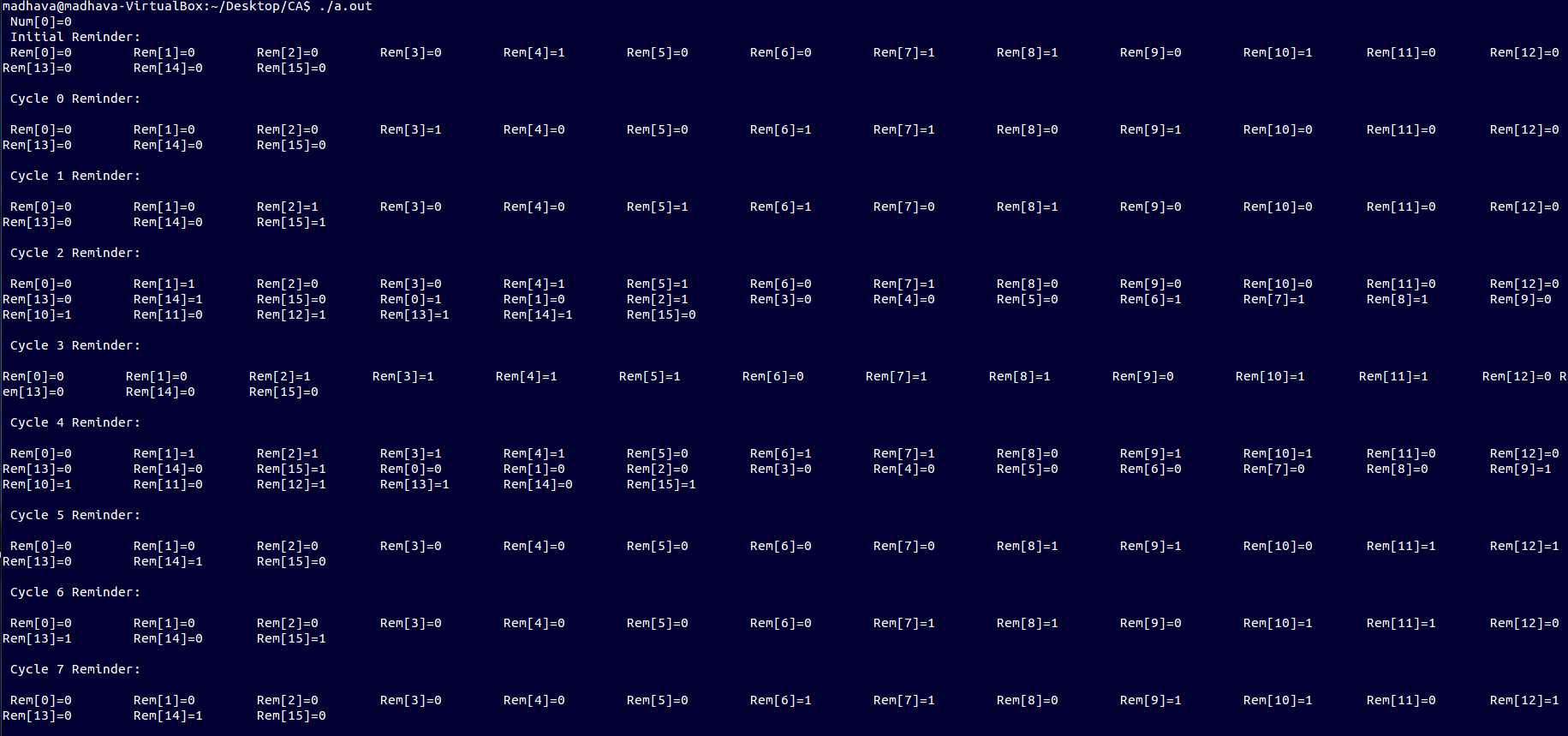
eff=(d+1.0)/avgaors;

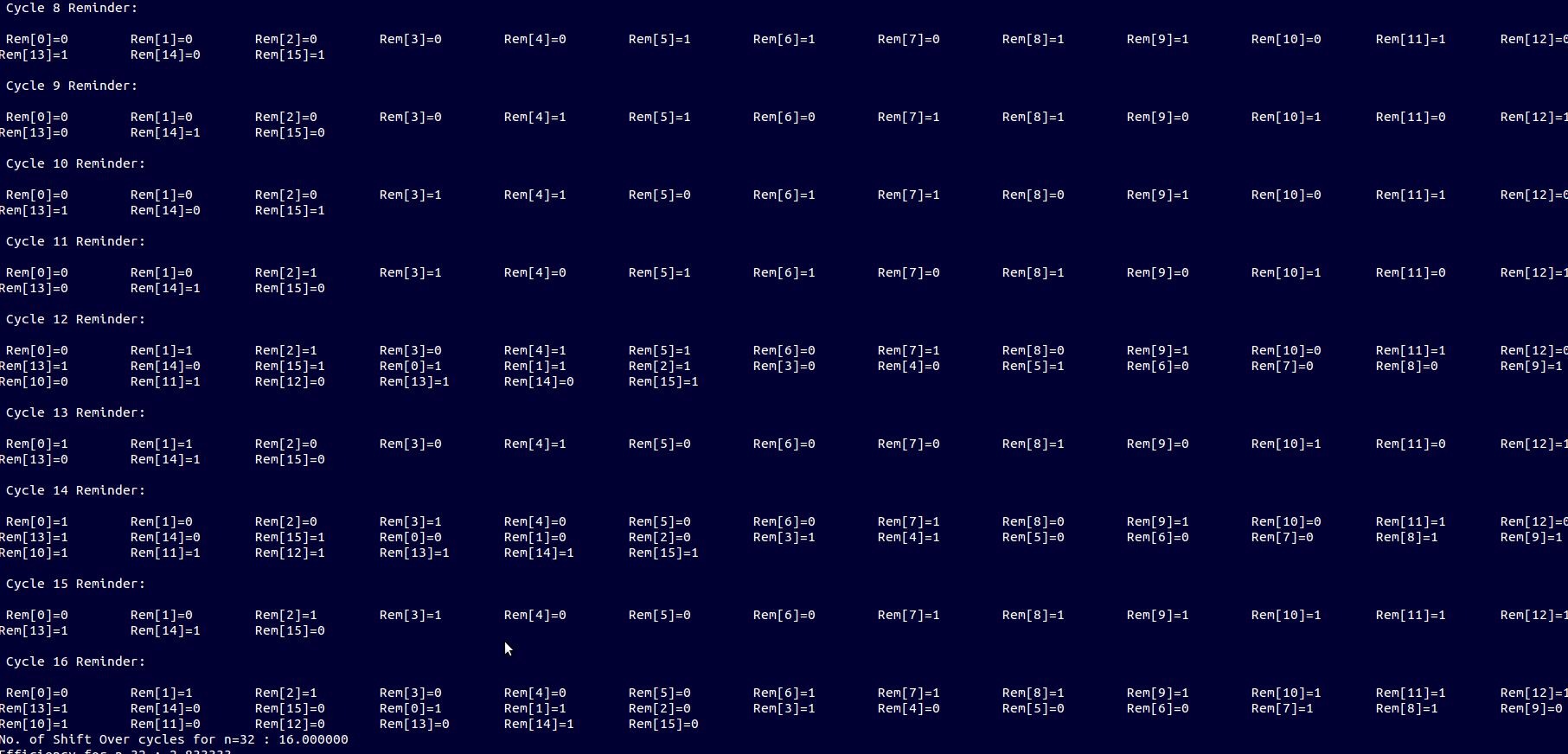
printf("\nEfficiency for n=%d : %f\n",n,eff);

}//For n

}

**Output:**

****

****