#include "reference.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define NUMBEHAVIORS 2

#define NUMPROCESSBEHAVIORS 4

int \_\_numprocesses=0;

int \_\_currentprocess;

unsigned long \_\_time=0;

unsigned long \_\_counter=0;

FILE \* \_\_file;

unsigned long \_\_lastref;

struct processData

{

unsigned int streams[8];

int numstreams;

int stride;

int numrefs;

unsigned int laststream[8];

unsigned long blocktime;

};

struct processData \_\_processData[256];

struct processBehavior

{

int numstreams;

int stride;

int numrefs;

int locality;

};

struct processBehavior \_\_processBehaviorData[][4] = {

{

{ 2, 1, 500, 0 },

{ 2, 1, 500, 0 },

{ 2, 1, 500, 0 },

{ 2, 1, 500, 0 },

},

{

{ 8, 1, 500, 4 },

{ 4, 1, 100, 4 },

{ 1, 1, 400, 4 },

{ 2, 1, 100, 0 },

},

};

void nextProcess();

void fault(int f)

{

if (!\_\_file)

{

\_\_file=fopen("output.txt","w");

}

if (f)

{

\_\_processData[\_\_currentprocess].blocktime=\_\_time+100000+rand()%50000;

nextProcess();

}

if (f)

{

fprintf(\_\_file,"%10ld Miss\n",\_\_lastref);

}

else

{

fprintf(\_\_file,"%10ld Hit\n",\_\_lastref);

}

}

void nextProcess()

{

int nextprocesstime;

int process;

// printf("Process Switch\n");

process=\_\_currentprocess;

\_\_currentprocess=(\_\_currentprocess+1)%\_\_numprocesses;

nextprocesstime=-1;

do

{

if (\_\_processData[\_\_currentprocess].blocktime<=\_\_time)

return;

else if (nextprocesstime == -1 || \_\_processData[\_\_currentprocess].blocktime

< \_\_processData[nextprocesstime].blocktime)

nextprocesstime=\_\_currentprocess;

\_\_currentprocess=(\_\_currentprocess+1)%\_\_numprocesses;

}

while (\_\_currentprocess!=process);

// printf("blocked Process\n");

\_\_currentprocess=nextprocesstime;

\_\_time=\_\_processData[\_\_currentprocess].blocktime;

}

void reference(struct ref \* theRef, int numProcesses, int data)

{

int stream;

theRef->operation=(rand()%3==2);

if (\_\_numprocesses==0)

{

int i,j,k;

\_\_numprocesses=numProcesses;

for (i=0; i<numProcesses; ++i)

{

int streambase;

\_\_processData[i].numstreams=

\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[i<NUMPROCESSBEHAVIORS ? i : NUMPROCESSBEHAVIORS-1].numstreams;

\_\_processData[i].stride=

\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[i<NUMPROCESSBEHAVIORS ? i : NUMPROCESSBEHAVIORS-1].stride;

for (j=0; j<\_\_processData[i].numstreams; ++j)

{

if (j==0)

{

streambase=rand()%32;

}

else

{

streambase+=(rand()%512)+1;

streambase=streambase%1024;

}

\_\_processData[i].laststream[j]=\_\_processData[i].streams[j]=streambase<<10;

\_\_processData[i].blocktime=0;

\_\_processData[i].numrefs=

\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[i<NUMPROCESSBEHAVIORS ? i : NUMPROCESSBEHAVIORS-1].numrefs/2+

rand()%\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[i<NUMPROCESSBEHAVIORS ? i : NUMPROCESSBEHAVIORS-1].numrefs;

// printf("Process %d Streambase %x NumRefs %d\n",i,streambase<<10,

// \_\_processData[i].numrefs);

}

}

\_\_currentprocess=rand()%\_\_numprocesses;

}

theRef->processid=\_\_currentprocess;

stream=rand()%\_\_processData[\_\_currentprocess].numstreams;

theRef->address=\_\_processData[\_\_currentprocess].laststream[stream]&0xfffff;

if (data==2)

{

theRef->address=(\_\_counter & 1) ?0x8000 : 0x2000;

++\_\_counter;

}

\_\_lastref=theRef->address;

switch(rand()%(

\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[\_\_currentprocess<NUMPROCESSBEHAVIORS ?

\_\_currentprocess : NUMPROCESSBEHAVIORS-1].locality+6))

{

case 0: case 1:

\_\_processData[\_\_currentprocess].laststream[stream]+=

(2<<(rand()%2))\*\_\_processData[\_\_currentprocess].stride;

break;

case 2:

\_\_processData[\_\_currentprocess].laststream[stream]+=

(4<<(rand()%4))\*\_\_processData[\_\_currentprocess].stride;

break;

case 3:

\_\_processData[\_\_currentprocess].laststream[stream]-=

(2<<(rand()%2))\*\_\_processData[\_\_currentprocess].stride;

break;

case 4:

\_\_processData[\_\_currentprocess].laststream[stream]=

\_\_processData[\_\_currentprocess].streams[stream]+((rand()%512)\*2);

break;

default:

// printf("Random\n");

\_\_processData[\_\_currentprocess].laststream[stream]=((rand()%0xffff)\*2);

break;

}

//printf("%d\n",\_\_time);

++\_\_time;

if (--\_\_processData[\_\_currentprocess].numrefs<=0)

{

\_\_processData[\_\_currentprocess].numrefs=

\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[\_\_currentprocess<NUMPROCESSBEHAVIORS ?

\_\_currentprocess : NUMPROCESSBEHAVIORS-1].numrefs/2+

rand()%\_\_processBehaviorData[data<NUMBEHAVIORS ? data : NUMBEHAVIORS-1]

[\_\_currentprocess<NUMPROCESSBEHAVIORS ? \_\_currentprocess :

NUMPROCESSBEHAVIORS-1].numrefs;

nextProcess();

}

}

/\*

int main()

{

struct ref theRef;

while (1)

{

reference(&theRef,10,1);;

printf("%x\n",theRef.address);

}

}

\*/