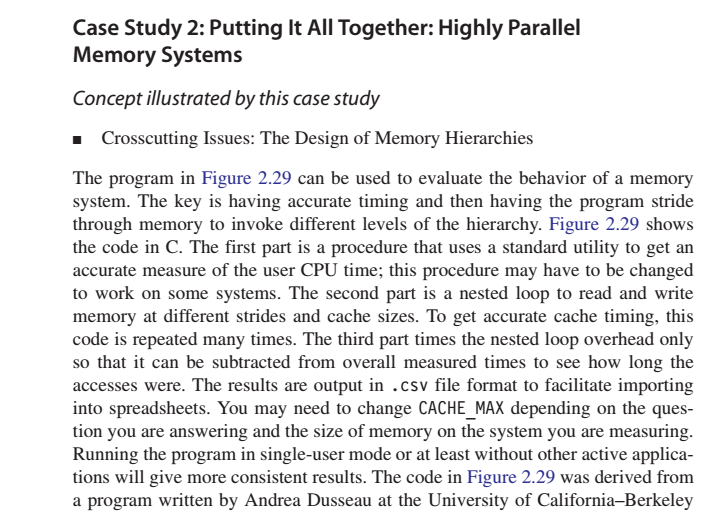
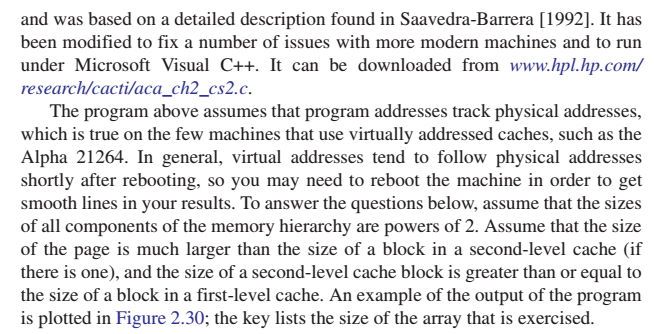
Assignment 5 in gem5.





**program for evaluating memory system.**

#include "stdafx.h"

#include <stdio.h>

#include <time.h>

#define ARRAY\_MIN (1024) /\* 1/4 smallest cache \*/

#define ARRAY\_MAX (4096\*4096) /\* 1/4 largest cache \*/

int x[ARRAY\_MAX]; /\* array going to stride through \*/

double get\_seconds() { /\* routine to read time in seconds \*/

\_\_time64\_t ltime;

\_time64( &ltime );

return (double) ltime;

}

int label(int i) {/\* generate text labels \*/

if (i<1e3) printf("%1dB,",i);

else if (i<1e6) printf("%1dK,",i/1024);

else if (i<1e9) printf("%1dM,",i/1048576);

else printf("%1dG,",i/1073741824);

return 0;

}

int \_tmain(int argc, \_TCHAR\* argv[]) {

int register nextstep, i, index, stride;

int csize;

double steps, tsteps;

double loadtime, lastsec, sec0, sec1, sec; /\* timing variables \*/

/\* Initialize output \*/

printf(" ,");

for (stride=1; stride <= ARRAY\_MAX/2; stride=stride\*2)

label(stride\*sizeof(int));

printf("\n");

/\* Main loop for each configuration \*/

for (csize=ARRAY\_MIN; csize <= ARRAY\_MAX; csize=csize\*2) {

label(csize\*sizeof(int)); /\* print cache size this loop \*/

for (stride=1; stride <= csize/2; stride=stride\*2) {

/\* Lay out path of memory references in array \*/

for (index=0; index < csize; index=index+stride)

x[index] = index + stride; /\* pointer to next \*/

x[index-stride] = 0; /\* loop back to beginning \*/

/\* Wait for timer to roll over \*/

lastsec = get\_seconds();

sec0 = get\_seconds(); while (sec0 == lastsec);

/\* Walk through path in array for twenty seconds \*/

/\* This gives 5% accuracy with second resolution \*/

steps = 0.0; /\* number of steps taken \*/

nextstep = 0; /\* start at beginning of path \*/

sec0 = get\_seconds(); /\* start timer \*/

{ /\* repeat until collect 20 seconds \*/

(i=stride;i!=0;i=i-1) { /\* keep samples same \*/

nextstep = 0;

do nextstep = x[nextstep]; /\* dependency \*/

while (nextstep != 0);

}

steps = steps + 1.0; /\* count loop iterations \*/

sec1 = get\_seconds(); /\* end timer \*/

} while ((sec1 - sec0) < 20.0); /\* collect 20 seconds \*/

sec = sec1 - sec0;

/\* Repeat empty loop to loop subtract overhead \*/

tsteps = 0.0; /\* used to match no. while iterations \*/

sec0 = get\_seconds(); /\* start timer \*/

{ /\* repeat until same no. iterations as above \*/

(i=stride;i!=0;i=i-1) { /\* keep samples same \*/

index = 0;

do index = index + stride;

while (index < csize);

}

tsteps = tsteps + 1.0;

sec1 = get\_seconds(); /\* - overhead \*/

} while (tsteps<steps); /\* until = no. iterations \*/

sec = sec - (sec1 - sec0);

loadtime = (sec\*1e9)/(steps\*csize);

/\* write out results in .csv format for Excel \*/

printf("%4.1f,", (loadtime<0.1) ? 0.1 : loadtime);

}; /\* end of inner for loop \*/

printf("\n");

}; /\* end of outer for loop \*/

return 0;

}

Attempt Exercise 2.4, page No: 135