作业：Memory、TLP

截止时间：12 月 28 日， 23:59

提交方式：email（guoych37@mail2.sysu.edu.cn ）

Q1: Assume there is a system with two processors (P0 and P1) sharing memory and each has its own private cache. Caches are organized as direct-mapped, and the coherence protocol is snooping-based MSI. At one moment, the snapshot of caches is as shown by the table below:

	Block #	Tag	Data word 1	Data word 2	Coherency State
P0	0	1000	10	20	M
	1	4000	500	600	S
	…				
	N	3000	2	4	S
P1	0	1000	10	10	I
	1	8000	500	600	S
	…				
	N	3000	2	4	S

a. For either P0 or P1, suppose the cache capacity is 16KB, and each block contains two words (as shown in the table, each word is of 4B), then please split the 32b address into tag-index-offset. Use the address notation *A[N:M]* to describe which address bits are used for each part.

b. If each core contains two cache levels, with access latency and hit ratio being 10/40 cycles and 85%/60%, respectively. Memory access latency is 100 clock cycles. Please calculate the average memory access time (AMAT).

c. If P0 is to update *Block 0*. What will happen?

d. If P1 is to update *Block 1*. Should *Block 1* on P0 be invalidated? Why or why not?

e. If P0 is to update *Block N*. How the coherency state on P0 and P1 will be changed?

f. If P1 brings in a block *M* for reading, and no cache has a copy, what state will *Block M* be in? Why?

g. Suppose the MSI protocol is extended to MESI. Will answer be the same for question e? Why or why not?

Q2: (2.4, P152) <2.6> Using the sample program results in Figure 2.33:

a. What are the overall size and block size of the second-level cache?

b. What is the miss penalty of the second-level cache?

c. What is the associativity of the second-level cache?

d. What is the size of the main memory?

e. What is the paging time if the page size is 4 KB?

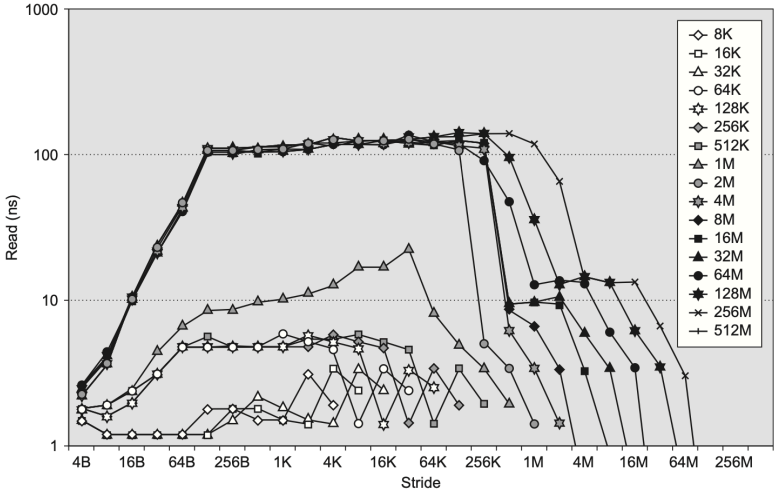

Figure 2.33 Sample results from program in Figure 2.32.

```
#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;
}
```

Figure 2.32 C program for evaluating memory system.

Q3: Please review the paper below following the required format.
Paper:
Nicolai Oswald, Vijay Nagarajan and Daniel J. Sorin Chen *et al.*, *HeteroGen: Automatic Synthesis of Heterogeneous Cache Coherence Protocols*, IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022.

链接：https://ieeexplore.ieee.org/abstract/document/9773254

Format:

A. Paper summary
Please provide a short summary of the paper that captures the key contributions.

==========

B. Key strengths and weaknesses
Please provide up to three strengths and three weaknesses, in the form of short (+) and (-) bullets, respectively.

==========

C. Comments to authors
Please provide detailed comments that support your above judgement, as well as constructive feedback to make the paper stronger. This should constitute the meat of your review.

Review 撰写参考：

[1]. O. Mutlu, Guidelines on Paper Reviews, https://course.ece.cmu.edu/~ece740/f13/lib/exe/fetch.php?media=onur-740-fall13-lecture0-3-how-to-do-the-paper-reviews.pdf

[2]. S. Krishnamurthi, How to Write Technical Paper Reviews, https://cs.brown.edu/~sk/Memos/Paper-Reviews/