Computer Architecture - prefetcher, Yu Liu , Oct 24th, 2019.

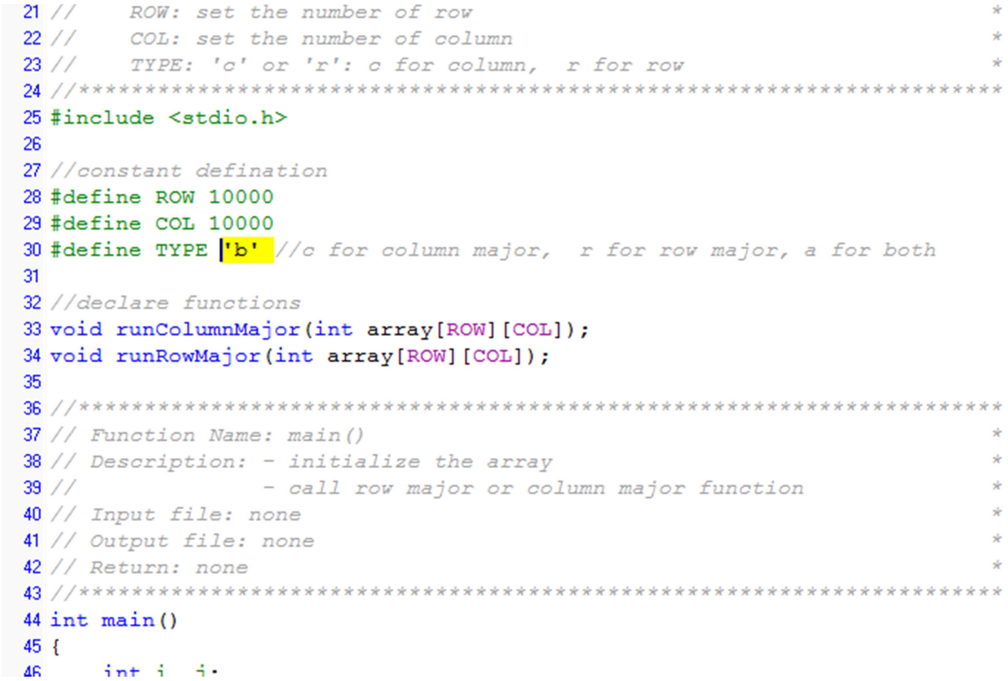
- These files in the package are for problem 2 and problem3.
- The source codes are under c language, including head are <stdio.h> , “stdbool.h” and <math.h>, can run under linux , unix, window 7 or above.
- Unzip the file of HW2_YULIU, put these files in one folder:
  - ✓ loop_yu.c
  - ✓ prefetcher_yu.c
  - ✓ README.pdf
  - ✓ stdbool.h
  - ✓ pinatrace_dhrystone.out (get it by extracting zip file)
  - ✓ pinatrace_linpack.out (get it by extracting zip file)

  result_dhrystone.txt and result_linpack.txt will be generated by running prefetcher_yu.c in the problem3.
- Below processes all run under linux as examples

Problem2(detect column-major or row-major memory layout for 2-dimentional array)**:**

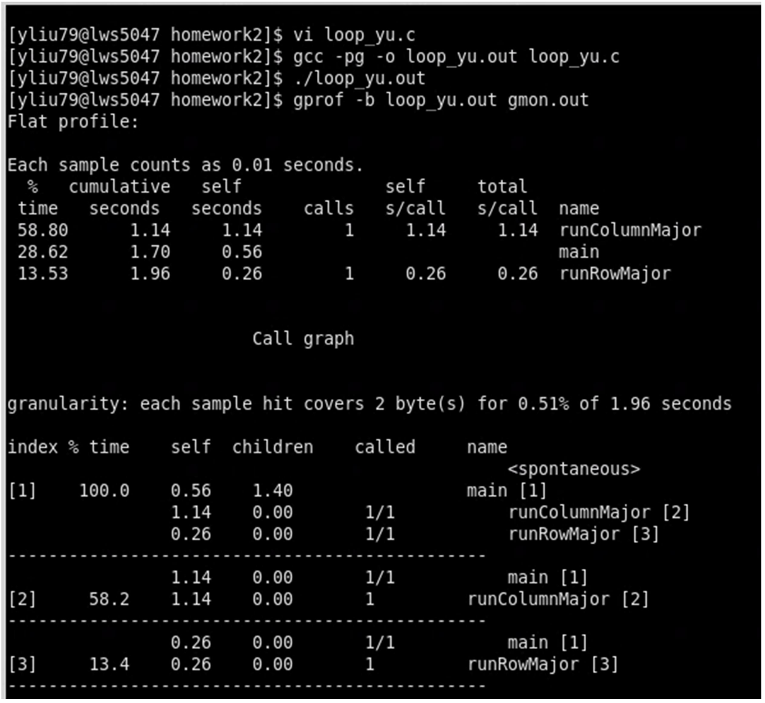
1. Use file of “loop_yu.c”
2. Configure the row-major or column-major by editing the file loop_yu.c in line 30. #define TYPE ‘b’, which means run both two types configuration, as below picture.

   $ vi loop_yu.c



```
21 //    ROW: set the number of row                                          *
22 //    COL: set the number of column                                       *
23 //    TYPE: 'c' or 'r': c for column,  r for row                          *
24 //***********************************************************************
25 #include <stdio.h>
26
27 //constant defination
28 #define ROW 10000
29 #define COL 10000
30 #define TYPE 'b'  //c for column major,  r for row major, a for both
31
32 //declare functions
33 void runColumnMajor(int array[ROW][COL]);
34 void runRowMajor(int array[ROW][COL]);
35
36 //***********************************************************************
37 // Function Name: main()                                                 *
38 // Description: - initialize the array                                   *
39 //              - call row major or column major function                *
40 // Input file: none                                                      *
41 // Output file: none                                                     *
42 // Return: none                                                          *
43 //***********************************************************************
44 int main()
45 {
46     int i, j;
```

3. Compile and run the program and then see the result, as below picture.
   $ gcc –pg –o loop_yu.out loop_yu.c
   $ ./loop_yu.out
   $ gropf –b loop_yu gmon.out



```
[yliu79@lws5047 homework2]$ vi loop_yu.c
[yliu79@lws5047 homework2]$ gcc -pg -o loop_yu.out loop_yu.c
[yliu79@lws5047 homework2]$ ./loop_yu.out
[yliu79@lws5047 homework2]$ gprof -b loop_yu.out gmon.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 58.80      1.14     1.14        1     1.14     1.14  runColumnMajor
 28.62      1.70     0.56                             main
 13.53      1.96     0.26        1     0.26     0.26  runRowMajor


                        Call graph


granularity: each sample hit covers 2 byte(s) for 0.51% of 1.96 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.56    1.40                 main [1]
                1.14    0.00       1/1           runColumnMajor [2]
                0.26    0.00       1/1           runRowMajor [3]
-----------------------------------------------
                1.14    0.00       1/1           main [1]
[2]     58.2    1.14    0.00       1         runColumnMajor [2]
-----------------------------------------------
                0.26    0.00       1/1           main [1]
[3]     13.4    0.26    0.00       1         runRowMajor [3]
-----------------------------------------------
```

4. Done

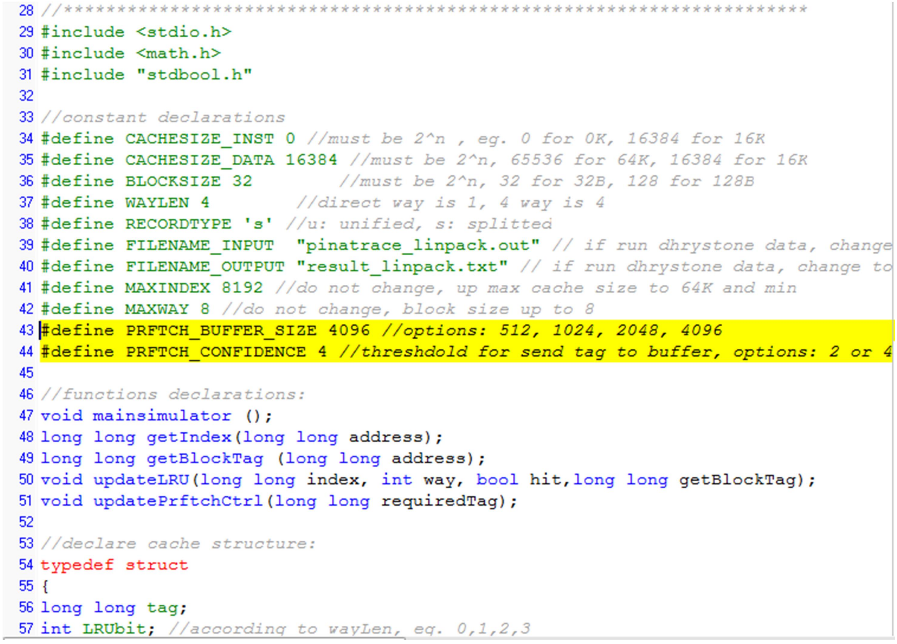
Problem 3 (run own programmed simulator with generated Pin file, under linux for example**) :**

1. Put source code and data in one folder.

- Prefeter_yu.c
- cachesim_pin.c
- stdbool.h
- pinatrace_dhrystone.out
- pinatrace_linpack.out

Configure the cache simulator parameter: edit prefetch_yu.c , select the input data source (eg. pinatrace_linpack.out), change the switch of prefetch buffer size and confidence. Options are 512, 1024, 2048, 4096 and 2,4. In line 39, 40, 43, 44
As below words and figure.

#define FILENAME_INPUT "pinatrace_linpack.out" // if run dhrystone data, change to "pinatrace_dhrysonte.out"
#define FILENAME_OUTPUT "result_linpack.txt" // if run dhrystone data, change to "result_dhrystone.txt"
#define PRFTCH_BUFFER_SIZE 4096 //options: 512, 1024, 2048, 4096
#define PRFTCH_CONFIDENCE 4 //threshdold for send tag to buffer, options: 2 or 4



```c
//*************************************************************************
#include <stdio.h>
#include <math.h>
#include "stdbool.h"

//constant declarations
#define CACHESIZE_INST 0 //must be 2^n , eg. 0 for 0K, 16384 for 16K
#define CACHESIZE_DATA 16384 //must be 2^n, 65536 for 64K, 16384 for 16K
#define BLOCKSIZE 32        //must be 2^n, 32 for 32B, 128 for 128B
#define WAYLEN 4         //direct way is 1, 4 way is 4
#define RECORDTYPE 's' //u: unified, s: splitted
#define FILENAME_INPUT  "pinatrace_linpack.out" // if run dhrystone data, change
#define FILENAME_OUTPUT "result_linpack.txt" // if run dhrystone data, change to
#define MAXINDEX 8192 //do not change, up max cache size to 64K and min
#define MAXWAY 8 //do not change, block size up to 8
#define PRFTCH_BUFFER_SIZE 4096 //options: 512, 1024, 2048, 4096
#define PRFTCH_CONFIDENCE 4 //threshdold for send tag to buffer, options: 2 or 4

//functions declarations:
void mainsimulator ();
long long getIndex(long long address);
long long getBlockTag (long long address);
void updateLRU(long long index, int way, bool hit,long long getBlockTag);
void updatePrftchCtrl(long long requiredTag);

//declare cache structure:
typedef struct
{
long long tag;
int LRUbit; //according to wayLen, eg. 0,1,2,3
```

1. Compile file of prefetcher_yu.c, use gcc command, add –lm for including <math.h>:

   ```
   $ gcc prefetcher_yu.c   –o   prefetcher_yu.out –lm
   ```

2. Run

   ```
   $ ./ prefetcher_yu.out
   ```

3. Output data

   A file named “result_dhrystone.txt” or “result_linpack.txt” will be generated automatically (samples already listed). Open and read the data inside.

4. Done