CENG311 Programming Assignment 4

Umut YILDIZ - 260201028

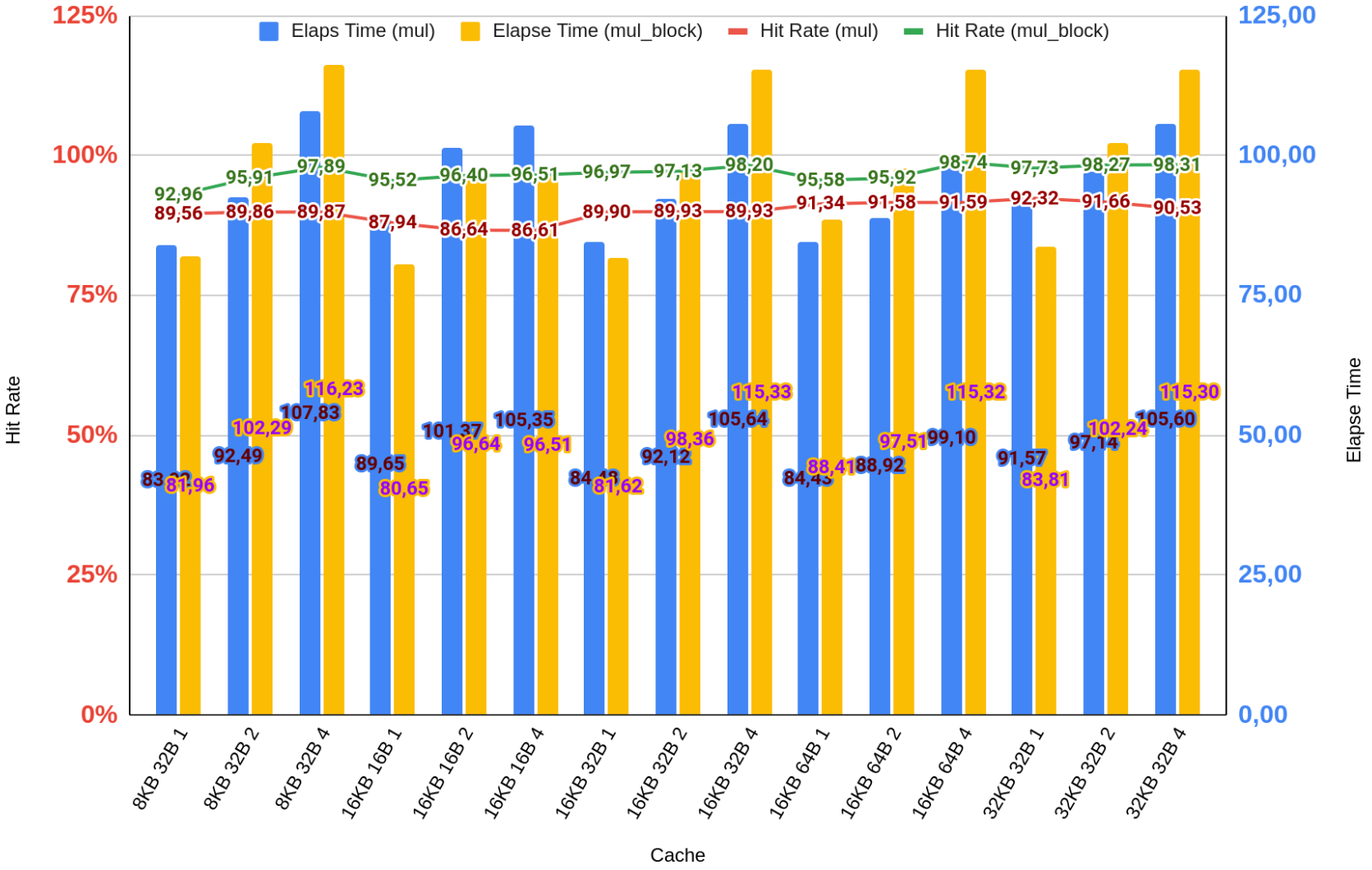
Ordinary Matrix Multiplication
vs
Blocked Matrix Multiplication

(examined by using “dcache” pin tool)

Most of us know that the blocked implementation is better than ordinary matrix multiplication. In this experiment I used “pin” application and its “dcache” tool to see what are the hit rates of some simulated caches. Here is the graph that represents the hit rate and elapsed time of the simulations with which cache attributes.

- Cache = L1 Size [KB], Block size [B], L1 Associativity
- This simulation can be re-created by following the last page instructions.
- This simulation is done one by one in order (allinone.sh). That is why the elapsed times are lower compared to the script written with “xterm”.

Cache	Hit Rate (mul)	Elapsed Time (mul)	Hit Rate (mul_block)	Elapsed Time (mul_block)
8KB 32B 1	89,56	83,92	92,96	81,96
8KB 32B 2	89,86	92,49	95,91	102,29
8KB 32B 4	89,87	107,83	97,89	116,23
16KB 16B 1	87,94	89,65	95,52	80,65
16KB 16B 2	86,64	101,37	96,40	96,64
16KB 16B 4	86,61	105,35	96,51	96,51
16KB 32B 1	89,90	84,48	96,97	81,62
16KB 32B 2	89,93	92,12	97,13	98,36
16KB 32B 4	89,93	105,64	98,20	115,33
16KB 64B 1	91,34	84,43	95,58	88,41
16KB 64B 2	91,58	88,92	95,92	97,51
16KB 64B 4	91,59	99,10	98,74	115,32
32KB 32B 1	92,32	91,57	97,73	83,81
32KB 32B 2	91,66	97,14	98,27	102,24
32KB 32B 4	90,53	105,60	98,31	115,30


Elaps Time (mul)
Elapse Time (mul_block)
Hit Rate (mul)
Hit Rate (mul_block)
Hit Rate
Elapse Time
Cache
125%
100%
75%
50%
25%
0%
125,00
100,00
75,00
50,00
25,00
0,00
8KB 32B 1
8KB 32B 2
8KB 32B 4
16KB 16B 1
16KB 16B 2
16KB 16B 4
16KB 32B 1
16KB 32B 2
16KB 32B 4
16KB 64B 1
16KB 64B 2
16KB 64B 4
32KB 32B 1
32KB 32B 2
32KB 32B 4
92,96
95,91
97,89
95,52
96,40
96,51
96,97
97,13
98,20
95,58
95,92
98,74
97,73
98,27
98,31
89,56
89,86
89,87
87,94
86,64
86,61
89,90
89,93
89,93
91,34
91,58
91,59
92,32
91,66
90,53
81,96
92,49
102,29
107,83
116,23
89,65
80,65
101,37
96,64
105,35
96,51
81,62
92,12
98,36
105,64
115,33
88,41
88,92
97,51
99,10
115,32
91,57
83,81
97,14
102,24
105,60
115,30

- As the Associativity increases the hit rate:

  1) When mul implementation is used the relationship is not clear. It mostly increases but sometimes it decreases.

  2) When block implementation is used, it increases significantly.

- Increase of Associativity mostly improves hit rate.

- As the Block Size increases the hit rate:

  1) When mul implementation is used, it is increasing.

  2) When block implementation is used, effects of block size changes are not stable.

  16KB-**16B**-1 => 16KB-**32B**-1: 95,52 => 96,97 (**increased**)
  16KB-**32B**-1 => 16KB-**64B**-1: 96,97 => 95,58 (**decreased**)

  16KB-**16B**-2 => 16KB-**32B**-2: 96,40 => 97,13 (**increased**)
  16KB-**32B**-2 => 16KB-**64B**-2: 97,13 => 95,92 (**decreased**)

  16KB-**16B**-4 => 16KB-**32B**-4: 96,51 => 98,20 (**increased**)
  16KB-**32B**-4 => 16KB-**64B**-4: 98,20 => 98,74 (**increased**)

- Increase of Block Size might improve hit rate depending on the cpu usage purpose (Some implementations might get worse).

- As the Cache Size increases the hit rate:

  1) When mul implementation is used, it is increasing.

  2) When block implementation is used, it is increasing.

- Increase of Block Size always improves the hit rate.

- Changes on these cache attributes affects the block implementation more.
- Elapsed times were not useful for this simulation. Because the elapsed times were measured by the c application not by the pin tool.

Instructions to re-create the simulation

1. Download the pin and unzip.
2. Decide which simulation script you will use
    - "**allinone.sh**" sets up the environment and starts the simulation one by one. It takes longer since the simulation will only use one core of your computer. This script takes ~50 minutes.
    - "**allinone_xterm.sh**" sets up the environment and creates 30 terminals to finish the simulation. Since there will be 30 processes each tries to finish its simulation, this method will last shorter but it will use much more resources of your computer. Also you need to install a program called "xterm". This can be installed by running "sudo apt install xterm". This script takes ~12 minutes.
3. Create a directory named hw where the pin executable is located.
4. Move the mul.c and the mul_block.c files inside the hw directory.
5. Make the shell script executable.
6. Run the shell script inside the directory where the pin executable is located.
7. After execution is finished there will be 30 result files inside the hw/out/ directory.