Assignment-2

(Roll Number: 20161076)

1)

- a) funct7() requires the most computing time
 - i) If funct7() is parallelized all these 260 calls can be ~ to single call and get significant drop in exclusive time of the callee funct8()

		1.31	16.00	13/26	funct6() [5]
		1.31	16.00	13/26	funct5() [2]
[3]	87.7	2.62	32.00	26	funct8() [3]
		26.17	0.00	260/260	funct7() [4]
		1.32	4.51	26/39	funct2() [6]
		26.17	0.00	260/260	funct8() [3]
[4]	66.3	26.17	0.00	260	funct7() [4]

- ii) Funct7() is called 260 times in total, all calls are from funct8() only
- iii) Funct7() does 260 calls funct8()

b)

9.33	29.84	3.67	5551	0.00	0.00	funct1()
7.92	32.96	3.12	3913	0.00	0.00	funct3()

Funct1() has highest number of calls, followed by funct3()

[6]	22.2	0.66 1.32 1.98 3.11	2.26 4.51 6.77 2.58	13/39 26/39 39 3900/3913	funct4() [9] funct8() [3] funct2() [6] funct3() [7]
		1.08	0.00	1638/5551	funct1() [8]
		0.01 3.11	0.01 2.58	13/3913 3900/3913	funct4() [9] funct2() [6]
[7]	14.5	3.12	2.59	3913	funct3() [7]
		2.59	0.00	3913/5551	funct1() [8]

		 0.66	2.26	 13/39	funct4() [9]
		1.32	4.51	26/39	funct8() [3]
[6]	22.2	1.98	6.77	39	funct2() [6]
		3.11	2.58	3900/3913	funct3() [7]
		1.08	0.00	1638/5551	funct1() [8]
		0.01	0.01	13/3913	funct4() [9]
		3.11	2.58	3900/3913	funct2() [6]
[7]	14.5	3.12	2.59	3913	funct3() [7]
		2.59	0.00	3913/5551	funct1() [8]
					funct1()

called 5551 times which has 1638 calls from funct2() and 3913 calls from funct3()

c) Funct5() is composed of 13 calls to funct6(), 13 calls to funct8() and 13 calls to funct4()

		1.31	38.07	13/13	main [1]
[2]	99.7	1.31	38.07	13	funct5() [2]
$ \cdot $		0.13	17.31	13/13	funct6() [5]
		1.31	16.00	13/26	funct8() [3]
		0.39	2.93	13/13	funct4() [9]

	Flat pr	ofile:						
	Fach sa	mple count	s as 0 01	seconds				
	A comment of the same	umulative	self	e seconds.	self	total		
	time	seconds	seconds	calls	s/call	s/call	name	
	66.46	26.17	26.17	260	0.10		funct7()	
	9.33	29.84	3.67	5551	0.00		funct1()	
	7.92 6.65	32.96 35.57	3.12 2.62	3913 26	0.00 0.10		funct3() funct8()	
	5.02	37.55	1.98	39	0.05		funct2()	
	3.34	38.86	1.31	13	0.10		funct5()	
	0.99	39.26	0.39	13	0.03		funct4()	
	0.33	39.39	0.13	13	0.01	1.34	funct6()	
	0.25	39.49 39.49	0.10 0.00	1	0.00	0.00	main _GLOBALsub_I	76funct1v
۱۱.	0.00	39.49	0.00	1	0.00			ization_and_destruction_0(int, int)
d)								
	index	% time	self	children	ı cal	led	name	
							<spontane< th=""><th>eous></th></spontane<>	eous>
	[1]	100.0	0.10	39.39			main [1]	
			1.31	38.07	13/	13	funct5()	[2]
			1.31	38.07	13/	'13	main [1]	
	[2]	99.7	1.31	38.07	13		funct5() [2]	
			0.13	17.31	13/	13	funct6()	
			1.31	16.00	13/	'26	funct8()	[3]
			0.39	2.93	13/	13	funct4()	[9]
			1.31	16.00	13/	'26	funct6()	[5]
			1.31	16.00	13/	'26	funct5()	[2]
	[3]	87.7	2.62	32.00	26		funct8() [3]	
			26.17	0.00	260/	260	funct7()	[4]
			1.32	4.51	26/	'39	funct2()	[6]
			26.17	0.00	260/	260	funct8()	[3]
	[4]	66.3	26.17	0.00	260		funct7() [4]	
	index	% time	self	children	ı cal	led	name	
							<spontane< th=""><th>eous></th></spontane<>	eous>
	[1]	100.0	0.10	39.39			main [1]	
			1.31	38.07	13/	13	funct5()	[2]
			1.31	38.07	13/	13	main [1]	
	[2]	99.7	1.31	38.07	13		funct5() [2]	
			0.13	17.31	13/		funct6()	
			1.31	16.00	13/		funct8()	
			0.39	2.93	13/	13	funct4()	[9]
			1.31	16.00	13/		funct6()	
			1.31	16.00	13/		funct5()	[2]
	[3]	87.7	2.62	32.00	26		funct8() [3]	
			26.17	0.00	260/		funct7()	
			1.32	4.51	26/	39	funct2()	[6]
			26.17	0.00	260/	260	funct8()	[3]
	[4]	66.3	26.17	0.00	260		funct7() [4]	

		0.13	17.31	13/13	funct5()	[2]
[5]	44.2	0.13			funct6() [5]	
		1.31	16.00	13/26	funct8()	[3]
		0.66	2.26	13/39	funct4()	[9]
		1.32	4.51	26/39	funct8()	[3]
[6]	22.2	1.98	6.77	39	funct2() [6]	
		3.11	2.58	3900/3913	funct3()	[7]
		1.08	0.00	1638/5551	funct1()	[8]
		0.01	0.01	13/3913	funct4()	[9]
		3.11	2.58	3900/3913	funct2()	[6]
[7]	14.5	3.12	2.59	3913	funct3() [7]	
		2.59	0.00	3913/5551		[8]
		1.08	0.00	1638/5551	funct2()	[6]
		2.59	0.00		funct3()	
[8]	9.3	3.67	0.00		funct1() [8]	
		0.39	2.93	13/13	funct5()	[2]
[9]	8.4	0.39	2.93		funct4() [9]	2-3
		0.66	2.26	13/39	funct2()	[6]
						L-3
				42.42		F22
		0.13		13/13	funct5()	[2]
[5]	44.2	0.13	17.31	13	funct6() [5]	
[5]	44.2					
[5]	44.2	0.13 1.31	17.31 16.00	13 13/26	funct6() [5] funct8()	[3]
[5]	44.2	0.13 1.31 0.66	17.31 16.00 2.26	13 13/26 13/39	funct6() [5] funct8() funct4()	[3]
		0.13 1.31 0.66 1.32	17.31 16.00 2.26 4.51	13 13/26 13/39 26/39	funct6() [5] funct8() funct4() funct8()	[3]
[5]	44.2	0.13 1.31 0.66 1.32 1.98	17.31 16.00 2.26 4.51 6.77	13 13/26 13/39 26/39 39	funct6() [5] funct8() funct4() funct8() funct2() [6]	[3] [9] [3]
		0.13 1.31 0.66 1.32 1.98	17.31 16.00 2.26 4.51 6.77	13 13/26 13/39 26/39 39	funct6() [5] funct8() funct4() funct8() funct2() [6] funct3()	[3] [9] [3]
		0.13 1.31 0.66 1.32 1.98	17.31 16.00 2.26 4.51 6.77	13 13/26 	funct6() [5] funct8() funct4() funct8() funct2() [6]	[3] [9] [3]
		0.13 1.31 0.66 1.32 1.98 3.11 1.08	17.31 16.00 2.26 4.51 6.77 2.58 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1()	[3] [9] [3] [7] [8]
		0.13 1.31 0.66 1.32 1.98 3.11	17.31 16.00 2.26 4.51 6.77 2.58 0.00	13 13/26 	funct6() [5] funct8() funct4() funct8() funct2() [6] funct3() funct1() funct4()	[3] [9] [3] [7] [8]
		0.13 1.31 0.66 1.32 1.98 3.11 1.08	17.31 16.00 	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2()	[3] [9] [3] [7] [8]
		0.13 1.31 0.66 1.32 1.98 3.11 1.08	17.31 16.00 2.26 4.51 6.77 2.58 0.00	13 13/26 	funct6() [5] funct8() funct4() funct8() funct2() [6] funct3() funct1() funct4()	[3] [9] [3] [7] [8]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08	17.31 16.00 	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2()	[3] [9] [3] [7] [8] [9] [6]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.11 3.12	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct3() [7] funct3()	[3] [9] [3] [7] [8] [9] [6]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.11 3.12	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct2() funct3() [7]	[3] [9] [3] [7] [8] [9] [6]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.11 3.12 2.59	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct3() [7] funct3()	[3] [9] [3] [7] [8] [9] [6] [8]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.11 3.12 2.59	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct3() [7] funct1() funct1()	[3] [9] [3] [7] [8] [9] [6] [8]
[7]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59 0.00 0.00 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct1() funct4() funct2() funct2() funct3() [7] funct1() funct1()	[3] [9] [3] [7] [8] [9] [6] [8]
[7]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59 0.00 0.00 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct1() funct4() funct2() funct2() funct3() [7] funct1() funct1()	[3] [9] [3] [7] [8] [9] [6] [8] [6] [7]
[7]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.11 3.12 2.59 1.08 2.59 3.67	17.31 16.00 2.26 4.51 6.77 2.58 0.00 0.01 2.58 2.59 0.00 0.00 0.00 0.00	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct3() [7] funct1() funct3() funct1() funct2() funct2() funct1() [8]	[3] [9] [3] [7] [8] [9] [6] [8] [6] [7]
[6]	22.2	0.13 1.31 0.66 1.32 1.98 3.11 1.08 0.01 3.12 2.59 1.08 2.59 3.67 0.39	17.31 16.00 	13 13/26 	funct6() [5] funct8() funct4() funct2() [6] funct3() funct1() funct4() funct2() funct3() [7] funct1() funct1() funct2() funct3() funct1() [8] funct5()	[3] [9] [3] [7] [8] [9] [6] [8] [6] [7] [2]

```
0.00
                                                __libc_csu_init [21]
                       0.00
[16]
               0.00
                       0.00
                                            _GLOBAL__sub_I__Z6funct1v [16]
                       0.00
                                  1/1
                                                __static_initialization_and_destruction_0(int, int) [17]
               0.00
                       0.00
                                                _GLOBAL__sub_I__Z6funct1v [16]
        0.0
               0.00
                       0.00
                                            __static_initialization_and_destruction_0(int, int) [17]
               0.00
                       0.00
                                                __libc_csu_init [21]
                                            _GLOBAL__sub_I__Z6funct1v [16]
[16]
               0.00
                       0.00
               0.00
                       0.00
                                                __static_initialization_and_destruction_0(int, int) [17]
               0.00
                       0.00
                                                _GLOBAL__sub_I__Z6funct1v [16]
        0.0
               0.00
                       0.00
                                             __static_initialization_and_destruction_0(int, int) [17]
```

2) Co1b.c is faster compared to co1a.c because of the extra variables that are defined in co1a. Average real time (in sec) for co1a.c 0.164 is and co1b.c is 0.110.

Sno	Co1a (in sec)	Co1b (in sec)
1	0.164	0.109
2	0.170	0.110
3	0.155	0.113
4	0.161	0.110
5	0.171	0.112

3) Co2b.c is faster compared to co2a.c because in co2b.c has sequential access to data where as co2a.c has non-sequential access to data so it is slower. Average real time (in sec) for co2a.c 1.5424 is and co2b.c is 0.2554.

Sno	Co2a (in sec)	Co2b (in sec)
1	1.469	0.281
2	1.625	0.248
3	1.627	0.252
4	1.499	0.247
5	1.492	0.249

- 4) Co4a Naïve Matrix Multiplication, co4b Transposed Matrix Multiplication, co4c Cache-Blocking Matrix Multiplication
 - a) Time taken by each variation of matrix multiplication vs size of matrix

Matrix	Co4a (in sec)	Co4b (in sec)	Co4c (in sec)
Size\Time			
1000	9.159	4.471	0.065
2000	1:27.63	35.246	0.244
4000	17:44.03	5:09.55	1.046

- b) From the table above it can be clearly understood that Cache-Blocking matrix multiplication (co4c) runs faster than other variants
- c) Likwid-profiling for cache misses

i) Naïve Matrix Multiplication

```
Group 1: L2CACHE
-----
       Event | Counter | Core 0 |
  INSTR_RETIRED_ANY | FIXC0 | 37037834796 |
 CPU_CLK_UNHALTED_CORE | FIXC1 | 25635164977 |
 CPU_CLK_UNHALTED_REF | FIXC2 | 20963793509 |
 L2_TRANS_ALL_REQUESTS | PMC0 | 1747906918 |
    L2_RQSTS_MISS | PMC1 | 180959783 |
 --------
      Metric | Core 0 |
 Runtime (RDTSC) [s] | 9.3124 |
 Runtime unhalted [s] | 11.1717 |
    Clock [MHz] | 2805.9606 |
               I 0.6921 I
       CPI
   L2 request rate | 0.0472 |
   L2 miss rate | 0.0049 |
    L2 miss ratio | 0.1035 |
```

ii) Transpose Matrix Multiplication

```
Group 1: L2CACHE
       Event | Counter | Core 0 |
INSTR_RETIRED_ANY | FIXC0 | 37074834883 |
 CPU_CLK_UNHALTED_CORE | FIXC1 | 12969594134 |
 CPU_CLK_UNHALTED_REF | FIXC2 | 10342022565 |
 L2_TRANS_ALL_REQUESTS | PMC0 | 593975335 |
   L2_RQSTS_MISS | PMC1 | 152988472 |
   -----+
     Metric | Core 0 |
 Runtime (RDTSC) [s] | 4.5868 |
 Runtime unhalted [s] | 5.6528 |
    Clock [MHz] | 2877.2850 |
       CPI
              0.3498 |
   L2 request rate | 0.0160 |
   L2 miss rate | 0.0041 |
   L2 miss ratio | 0.2576 |
```

iii) Cache-Blocking Matrix Multiplication