# OpenBLAS test manual

## Platform and Architecture

This manual is based on the following platforms and configurations:

ARM platform:

- Architecture: ARM(aarch64)

- Server version: Kunpeng 920-7260

- Kernel version: Linux kunpeng2 (5.10.0-209.0.0.117.oe2203sp3.aarch64 #1 SMP Wed Jun 19 18:00:16 CST 2024)

- GCC&G++ version: 10.3.0

- JAVA version: 1.8.0\_412

- Python version: 3.12.2

RISC-V Platform:

- Architecture: RISCV (RISC-V64)

- Server version: SG2042

- Kernel version: Linux openeuler-riscv-4-2 6.6.0 #1 SMP Tue Jul 2 11:21:06 CST 2024 riscv64 riscv64 riscv64 GNU/Linux

- GCC&G++ version: 12.3.1

- JAVA version: 11.0.22

- Python version: 3.12.2

## test

### Build executable file

We need to download the benchmark framework and execute the script directly:

```sh

./build.sh

```

### Test case 1: benchmark\_axpby\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpby\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 146.44 MFlops 0.000010 sec

```

### Test case 2: benchmark\_iamin\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamin\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 88.99 MFlops 0.000009 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 3: benchmark\_spr\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops Time

100 : 1628.00 MFlops 0.000025 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 4: benchmark\_trsv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Transa = 'N' Inc\_x = 1 uplo=L diag=U loop = 1

SIZE Flops

============================================

100 : 413.05 MFlops 0.000024 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 5: benchmark\_gemm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step=10 : Transa=N : Transb=N

SIZE Flops Time

M= 100, N= 100, K= 100: 5517.66 MFlops 0.001450 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 6: benchmark\_spmv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spmv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 2193.58 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 7: benchmark\_gemm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step=10 : Transa=N : Transb=N

SIZE Flops Time

M= 100, N= 100, K= 100: 6048.33 MFlops 0.000331 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 8: benchmark\_trmm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U

SIZE Flops

100: 2929.46 MFlops 0.000341 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 9: benchmark\_axpy\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpy\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 39.45 MFlops 0.000005070 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 10: benchmark\_spmv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spmv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 666.22 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 11: benchmark\_hpmv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hpmv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 412.46 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 12: benchmark\_zdot\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_zdot\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 89.09 MFlops 0.000009 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 13: benchmark\_min\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_min\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 167.71 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 14: benchmark\_amin\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amin\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 166.32 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 15: benchmark\_syr2k

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2k 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 7360.25 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 16: benchmark\_dot\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_dot\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100: 41.93 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 17: benchmark\_max

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_max 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 71.05 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 18: benchmark\_gemm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step=10 : Transa=N : Transb=N

SIZE Flops Time

M= 100, N= 100, K= 100: 20014.51 MFlops 0.000400 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 19: benchmark\_nrm2\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_nrm2\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 36.23 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 20: benchmark\_rot\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rot\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 214.67 MFlops 0.000011 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 21: benchmark\_syr2k\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2k\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 13881.42 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 22: benchmark\_ger\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_ger\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 3695.15 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 23: benchmark\_gesv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gesv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Flops Time

100x100: 4913.05 MFlops 0.002171 s

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 24: benchmark\_amin\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amin\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 81.47 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 25: benchmark\_potrf\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_potrf\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U

100: 3289.26 MFlops: 0.000 Sec: Test=F

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 26: benchmark\_getri\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_getri\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE FLops Time Lwork

100: 4582.91 MFlops: 0.00 Sec: 6400

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 27: benchmark\_syr\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops

100 : 406.50 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 28: benchmark\_hpmv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hpmv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 383.22 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 29: benchmark\_axpby

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpby 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 41.90 MFlops 0.000007 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 30: benchmark\_iamax

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamax 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 70.55 MBytes 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 31: benchmark\_syrk\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syrk\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Loops = 1

SIZE Flops

100 : 5768.01 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 32: benchmark\_gesv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gesv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Flops Time

100x100: 10001.19 MFlops 0.001067 s

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 33: benchmark\_gemm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemm\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step=10 : Transa=N : Transb=N

SIZE Flops Time

M= 100, N= 100, K= 100: 7048.95 MFlops 0.000284 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 34: benchmark\_spr

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops Time

100 : 447.23 MFlops 0.000022 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 35: benchmark\_scal\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_scal\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 20.12 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 36: benchmark\_tpsv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpsv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 352.98 MFlops 0.000028330 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 37: benchmark\_spmv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spmv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 2252.25 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 38: benchmark\_swap

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_swap 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 75.19 MBytes 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 39: benchmark\_syr2\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops

100 : 658.98 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 40: benchmark\_rotm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rotm\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 123.71 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 41: benchmark\_hemm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 6330.82 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 42: benchmark\_imax\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_imax\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 159.36 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 43: benchmark\_spr\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops Time

100: 1820.66 MFlops 0.000022 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 44: benchmark\_copy\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_copy\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 169.13 MBytes 0.000004730 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 45: benchmark\_geev\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_geev\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Job=V

SIZE FLops Time Lwork

100: 2901.59 MFlops: 0.04 Sec: 13000

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 46: benchmark\_potrf

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_potrf 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U

100: 989.36 MFlops: 0.000 Sec: Test=F

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 47: benchmark\_swap\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_swap\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 75.97 MBytes 0.000011 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 48: benchmark\_imin

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_imin 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 71.56 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 49: benchmark\_rotm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rotm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100: 519.48 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 50: benchmark\_axpy\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpy\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 71.24 MFlops 0.000011230 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 51: benchmark\_symm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 12242.71 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 52: benchmark\_gemv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Trans = 'N' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100: 3838.77 MFlops 0.000021 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 53: benchmark\_hemm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 12449.62 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 54: benchmark\_trmv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 1789.71 MFlops 0.000022350 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 55: benchmark\_symm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symm\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 7277.23 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 56: benchmark\_symv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1212.12 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 57: benchmark\_her2k\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2k\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 5388.73 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 58: benchmark\_cholesky\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_cholesky\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

M = 100: 1.776357e-15 1556.133 MFlops 1.776357e-15 4111.678 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 59: benchmark\_tpmv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpmv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 376.65 MFlops 0.000026550 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 60: benchmark\_rot

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rot 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 102.39 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 61: benchmark\_tpsv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpsv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 384.91 MFlops 0.000025980 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 62: benchmark\_asum

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_asum 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 37.45 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 63: benchmark\_tpsv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpsv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 1126.44 MFlops 0.000035510 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 64: benchmark\_imin\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_imin\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 168.42 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 65: benchmark\_hpmv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hpmv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1717.48 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 66: benchmark\_ger\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_ger\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 2489.11 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 67: benchmark\_rot\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rot\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 126.05 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 68: benchmark\_getri\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_getri\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE FLops Time Lwork

100: 9796.27 MFlops: 0.00 Sec: 6400

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 69: benchmark\_zdot\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_zdot\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 71.17 MFlops 0.000011 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 70: benchmark\_tpsv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpsv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 834.72 MFlops 0.000047920 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 71: benchmark\_geev\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_geev\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Job=V

SIZE FLops Time Lwork

100: 4414.61 MFlops: 0.02 Sec: 13000

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 72: benchmark\_gemv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Trans = 'N' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100: 2026.34 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 73: benchmark\_trsm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U Loops = 1

SIZE Flops

100 : 2423.13 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 74: benchmark\_hemm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 4415.60 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 75: benchmark\_nrm2

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_nrm2 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 35.71 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 76: benchmark\_asum\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_asum\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 43.43 MFlops 0.000009 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 77: benchmark\_asum\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_asum\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 37.81 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 78: benchmark\_linpack\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_linpack\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Residual Decompose Solve Total

100: 1.417197e-06 7386.27 MFlops 1294.50 MFlops 6495.91 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 79: benchmark\_trsv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Transa = 'N' Inc\_x = 1 uplo=L diag=U loop = 1

SIZE Flops

============================================

100 : 287.52 MFlops 0.000035 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 80: benchmark\_cholesky\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_cholesky\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

M = 100: 1.776357e-15 2315.316 MFlops 1.776357e-15 5046.964 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 81: benchmark\_tpmv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpmv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100: 927.64 MFlops 0.000043120 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 82: benchmark\_dot

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_dot 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 32.79 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 83: benchmark\_syr\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops

100 : 1493.09 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 84: benchmark\_symv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 792.71 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 85: benchmark\_her2k

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2k 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 3041.32 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 86: benchmark\_syr

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops

100 : 431.41 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 87: benchmark\_trmv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 348.80 MFlops 0.000028670 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 88: benchmark\_nrm2\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_nrm2\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 73.06 MFlops 0.000011 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 89: benchmark\_zdot\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_zdot\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 16.98 MFlops 0.000012 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 90: benchmark\_iamax\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamax\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 81.05 MBytes 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 91: benchmark\_swap\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_swap\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 154.14 MBytes 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 92: benchmark\_scal

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_scal 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 18.28 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 93: benchmark\_symv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 649.77 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 94: benchmark\_hbmv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hbmv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' k = 1 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 17.62 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 95: benchmark\_syr2\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops

100 : 3133.57 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 96: benchmark\_swap\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_swap\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 125.10 MBytes 0.000013 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 97: benchmark\_getri\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_getri\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE FLops Time Lwork

100: 4508.45 MFlops: 0.00 Sec: 6400

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 98: benchmark\_linpack

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_linpack 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Residual Decompose Solve Total

100: 7.653236e-05 1918.63 MFlops 1136.36 MFlops 1880.92 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 99: benchmark\_copy

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_copy 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 79.05 MBytes 0.000005060 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 100: benchmark\_tpmv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpmv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100: 1098.30 MFlops 0.000036420 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 101: benchmark\_rotm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rotm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 3428.57 MFlops 0.000001 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 102: benchmark\_hemv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 472.03 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 103: benchmark\_potrf\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_potrf\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U

100: 2621.80 MFlops: 0.001 Sec: Test=F

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 104: benchmark\_zdot

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_zdot 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 20.02 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 105: benchmark\_amax

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amax 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 14.70 MFlops 0.000027 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 106: benchmark\_geev\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_geev\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Job=V

SIZE FLops Time Lwork

100: 2412.11 MFlops: 0.01 Sec: 13000

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 107: benchmark\_getri

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_getri 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE FLops Time Lwork

100: 7216.86 MFlops: 0.00 Sec: 6400

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 108: benchmark\_symv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1394.70 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 109: benchmark\_potrf\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_potrf\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U

100: 1138.61 MFlops: 0.000 Sec: Test=F

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 110: benchmark\_axpby\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpby\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 121.95 MFlops 0.000011 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 111: benchmark\_gemv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Trans = 'N' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100: 2659.57 MFlops 0.000008 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 112: benchmark\_iamin\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamin\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 161.62 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 113: benchmark\_hbmv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hbmv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' k = 1 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 83.62 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 114: benchmark\_ger\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_ger\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1777.78 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 115: benchmark\_rotm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rotm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 857.14 MFlops 0.000001 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 116: benchmark\_dot\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_dot\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 120.85 MFlops 0.000007 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 117: benchmark\_syr2

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops

100 : 778.21 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 118: benchmark\_ger

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_ger 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 2418.38 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 119: benchmark\_herk\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_herk\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 8765.97 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 120: benchmark\_her\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 1204.09 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 121: benchmark\_her2

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 309.60 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 122: benchmark\_scal\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_scal\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 51.19 MFlops 0.000012 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 123: benchmark\_trsm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U Loops = 1

SIZE Flops

100 : 4104.50 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 124: benchmark\_amax\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amax\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 170.21 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 125: benchmark\_spr\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops Time

100: 423.01 MFlops 0.000024 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 126: benchmark\_amin

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amin 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 61.92 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 127: benchmark\_cholesky\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_cholesky\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

M = 100: 9.536743e-07 7049.223 MFlops 9.536743e-07 9615.605 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 128: benchmark\_linpack\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_linpack\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Residual Decompose Solve Total

100: 1.978507e-14 3258.95 MFlops 894.95 MFlops 3026.13 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 129: benchmark\_axpy\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpy\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 81.63 MFlops 0.000009800 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 130: benchmark\_spr2\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr2\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops Time

100: 2792.32 MFlops 0.000029 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 131: benchmark\_trmv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 1188.00 MFlops 0.000033670 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 132: benchmark\_syrk

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syrk 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Loops = 1

SIZE Flops

100 : 2947.33 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 133: benchmark\_trmv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100 : 395.26 MFlops 0.000025300 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 134: benchmark\_asum\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_asum\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 40.86 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 135: benchmark\_min

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_min 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 74.63 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 136: benchmark\_axpby\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpby\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 65.79 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 137: benchmark\_max\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_max\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 145.45 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 138: benchmark\_trsm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsm\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U Loops = 1

SIZE Flops

100 : 4066.53 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 139: benchmark\_amax\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amax\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 4.44 MFlops 0.000180 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 140: benchmark\_symm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 5429.07 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 141: benchmark\_her\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 1095.89 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 142: benchmark\_dot\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_dot\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 169.13 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 143: benchmark\_trsv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Transa = 'N' Inc\_x = 1 uplo=L diag=U loop = 1

SIZE Flops

============================================

100 : 175.22 MFlops 0.000057 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 144: benchmark\_her2k\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2k\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 18005.04 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 145: benchmark\_imax

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_imax 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 69.81 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 146: benchmark\_trsv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Transa = 'N' Inc\_x = 1 uplo=L diag=U loop = 1

SIZE Flops

============================================

100 : 347.71 MFlops 0.000029 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 147: benchmark\_iamax\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamax\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 169.13 MBytes 0.000009 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 148: benchmark\_herk\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_herk\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 1067.97 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 149: benchmark\_trmm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U

SIZE Flops

100 : 7661.81 MFlops 0.000522 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 150: benchmark\_iamin\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamin\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 162.27 MFlops 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 151: benchmark\_syr2k\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2k\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 4033.56 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 152: benchmark\_trmm\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmm\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U

SIZE Flops

100: 5119.41 MFlops 0.000781 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 153: benchmark\_hbmv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hbmv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' k = 1 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 71.81 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 154: benchmark\_linpack\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_linpack\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Residual Decompose Solve Total

100: 2.486900e-14 2988.73 MFlops 1324.50 MFlops 2883.22 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 155: benchmark\_copy\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_copy\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100: 158.42 MBytes 0.000010100 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 156: benchmark\_iamax\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamax\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 148.42MBytes 0.000005 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 157: benchmark\_trsm\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trsm\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U Loops = 1

SIZE Flops

100 : 6740.93 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 158: benchmark\_scal\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_scal\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 62.37 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 159: benchmark\_tpmv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_tpmv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Diag = U Loops=1 Inc\_x=1

SIZE Flops

100: 376.51 MFlops 0.000026560 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 160: benchmark\_her2\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 1393.73 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 161: benchmark\_spmv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spmv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 620.73 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 162: benchmark\_syr2\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops

100 : 2663.12 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 163: benchmark\_syrk\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syrk\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Loops = 1

SIZE Flops

100 : 2516.42 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 164: benchmark\_symm

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_symm 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U

SIZE Flops

100 : 6036.82 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 165: benchmark\_gemv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gemv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Trans = 'N' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100: 1710.50 MFlops 0.000047 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 166: benchmark\_herk\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_herk\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 4676.56 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 167: benchmark\_axpy

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_axpy 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 33.50 MFlops 0.000005970 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 168: benchmark\_amin\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amin\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 165.98 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 169: benchmark\_gesv

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gesv 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Flops Time

100x100: 4673.61 MFlops 0.000571 s

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 170: benchmark\_syr2k\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr2k\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 5294.61 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 171: benchmark\_geev

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_geev 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Job=V

SIZE FLops Time Lwork

100: 3237.22 MFlops: 0.01 Sec: 13000

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 172: benchmark\_spr2\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr2\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops Time

100: 692.28 MFlops 0.000029 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 173: benchmark\_hpmv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hpmv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1950.27 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 174: benchmark\_amax\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_amax\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 27.82 MFlops 0.000058 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 175: benchmark\_iamin

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_iamin 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 72.20 MFlops 0.000006 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 176: benchmark\_syr\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syr\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1

SIZE Flops

100 : 1665.28 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 177: benchmark\_syrk\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_syrk\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N Loops = 1

SIZE Flops

100 : 13897.09 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 178: benchmark\_trmm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_trmm\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Side = L Uplo = U Trans = N Diag = U

SIZE Flops

100 : 2444.51 MFlops 0.000409 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 179: benchmark\_copy\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_copy\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 83.68 MBytes 0.000009560 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 180: benchmark\_nrm2\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_nrm2\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100 : 60.88 MFlops 0.000013 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 181: benchmark\_her2\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 2132.20 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 182: benchmark\_hemv\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemv\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 2111.93 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 183: benchmark\_hemv\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemv\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 1461.99 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 184: benchmark\_gesv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_gesv\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

SIZE Flops Time

100x100: 4625.05 MFlops 0.000577 s

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 185: benchmark\_her

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Trans = N

SIZE Flops

100 : 221.43 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 186: benchmark\_spr2

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr2 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops Time

100 : 818.00 MFlops 0.000024 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 187: benchmark\_cholesky

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_cholesky 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10

M = 100: 9.536743e-07 878.694 MFlops 9.536743e-07 5803.602 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 188: benchmark\_rot\_COMPLEX

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_rot\_COMPLEX 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100 : 244.40 MFlops 0.000010 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 189: benchmark\_spr2\_COMPLEX\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_spr2\_COMPLEX\_DOUBLE 100 100 10

```

The running results are:

```sh

From : 100 To : 100 Step = 10 Uplo = U Inc\_x = 1 Inc\_y = 1

SIZE Flops Time

100: 2593.19 MFlops 0.000031 sec

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 190: benchmark\_hemv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemv\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = 'L' Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 307.46 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 191: benchmark\_hbmv\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hbmv\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = 'L' k = 1 Inc\_x = 1 Inc\_y = 1 Loops = 1

SIZE Flops

100x100 : 14.14 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 192: benchmark\_hemm\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_hemm\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Side = L Uplo = U

SIZE Flops

100 : 1499.53 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 193: benchmark\_her2\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = U Trans = N

SIZE Flops

100 : 298.28 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 194: benchmark\_her2k\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her2k\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = U Trans = N

SIZE Flops

100 : 1281.84 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 195: benchmark\_her\_DOUBLE

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_her\_DOUBLE 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = U Trans = N

SIZE Flops

100 : 166.61 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

### Test case 196: benchmark\_herk

After executing `build.sh`, go to the `exe` directory and run the test case:

```sh

./benchmark\_herk 100 200 101

```

The running results are:

```sh

From : 100 To : 200 Step = 101 Uplo = U Trans = N

SIZE Flops

100 : 1982.87 MFlops

```

The Flops and sec of ARM platform and RISCV platform will be different.

## perf automated script test

For the test SIZE, since OpenBLAS is a linear algebra library, each test item is essentially a mathematical operation, such as matrix transposition or matrix inversion, which involves the issue of data size. Here, SIZE=100 is executed uniformly, only once, and the test starts:

```bash

./openblas.sh

```

### Test result file description

First is `run\_result`, which stores the output of all successfully run tests, such as:

```

From : 100 To : 100 Step = 10 Inc\_x = 1 Loops = 1

SIZE Flops

100: 106.95 MFlops 0.000007 sec

```

This is the output of the executable file, indicating the parameters such as SIZE and the total time and Flops required to perform this mathematical operation.

Then there is the folder `test\_result` folder, which saves all the results of the perf command output, such as:

```

# started on Sat Aug 3 16:45:32 2024

Performance counter stats for './exe/benchmark\_amax\_DOUBLE 100 100 10':

2,866,460 ns duration\_time: # 1.184 G/sec

2.42 msec task-clock: # 0.845 CPUs utilized

3,865,711 cycles: # 1.597 GHz

4,999,999 instructions: # 1.29 insn per cycle

2,123,444 cache-references: # 877.163 M/sec

44,382 cache-misses: # 2.09% of all cache refs

971,804 branches: # 401.438 M/sec

31,641 branch-misses: # 3.26% of all branches

2,123,444 L1-dcache-loads: # 877.163 M/sec

44,382 L1-dcache-load-misses: # 2.09% of all L1-dcache accesses

3,349 LLC-load-misses: # 12.74% of all L1-icache accesses

26,283 LLC-loads: # 10.857 M/sec

0.002866460 seconds time elapsed

0.000000000 seconds user

0.002918000 seconds sys

```

This is also what will be printed on the terminal if you use the `perf xxxx....` command in the script.

At this point, all the tests are finished and the required performance data are in the various txt files in the folder.

### Result formatting

The test results are saved in a txt file. Here is another script to format the results of the txt file into an xlsx file to match the format in Tencent Documents.

First, open the virtual environment or call the system environment's Python to ensure that there are dependencies:

```sh

pip install pandas openpyxl

```

Then you need to select the corresponding regular match according to the perf version, because the perf output data may be a number such as `2222120` without `,`, or it may output a number such as `2,222,120`, so you need to match them separately.

Then execute the script:

```sh

python xlsx.py

```

Then an xlsx file `result.xlsx` will be generated in the current directory.