```cpp
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cfloat>
#include <csignal>
#include <fenv.h>

/* Manually alter mask value on stack
 * This will work only on 32-bit system */

void handler(int signum, siginfo_t *siginfo, void *context) {
    unsigned int base;
    unsigned short *control;

    base = (unsigned int)&base; // get data structure on stack
    control = (unsigned short*)*(unsigned int*)(base + 260); // get value ptr on stack

    if(siginfo->si_code == 6) { // turn on mask for UNDERFLOW and INEXACT
        *control |= (FE_UNDERFLOW | FE_INEXACT);
    }
    else if(siginfo->si_code == 4) { // turn on mask for OVERFLOW
        *control |= FE_OVERFLOW;
    }

    fprintf(stderr, "fp exception %x at address %x\n", siginfo->si_code, siginfo->si_addr);
    return;
}

int main() {
    // turn off mask for all float exception
    #pragma STDC FENV_ACCESS ON
    feenableexcept(FE_DIVBYZERO | FE_INEXACT | FE_INVALID | FE_OVERFLOW | FE_UNDERFLOW);

    // register sigaction
    struct sigaction act;
    memset(&act, 0, sizeof(act));

    act.sa_sigaction = &handler;
    act.sa_flags = SA_SIGINFO;

    if (sigaction(SIGFPE, &act, NULL) < 0) {
        fprintf(stderr, "Set sigaction failed.\n");
        exit(EXIT_FAILURE);
    }

    // start calculation
    double x;
    x = DBL_MIN;
    printf("min_normal = %g\n", x);

    __asm("mov $0x12345678,%esi");
    x = x / 13.0;
    printf("min_normal / 13.0 = %g\n", x);

    x = DBL_MAX;
    printf("max_normal = %g\n", x);

    x = x * x;
    printf("max_normal * max_normal = %g\n", x);

    return 0;
}
```

```
min_normal = 2.22507e-308
fp exception 6 at address 8048853
min_normal / 13.0 = 1.7116e-309
max_normal = 1.79769e+308
fp exception 4 at address 8048895
max_normal * max_normal = 1.79769e+308
```

```
function main()
n = 500;
M = rand(n, n);
M = M * M' + eye;

% One-for version
fprintf('One-for version\n');
t = cputime; tic;
A = M;
for k = 1: n
    A(k:n, k) = A(k:n, k) / sqrt(A(k, k));
    A(k, k + 1:n) = 0;
    A(k + 1:n, k + 1:n) = A(k + 1:n, k + 1:n) - A(k + 1:n, k) * A(k + 1:n, k)';
end
fprintf('CPU time used: %f\n', cputime - t);
fprintf('Elapsed time used: %f\n\n', toc);

% Two-for version
fprintf('Two-for version\n');
t = cputime; tic;
A = M;
for k = 1:n
    A(k:n, k) = A(k:n, k) / sqrt(A(k, k));
    A(k, k + 1:n) = 0;
    for j = k + 1:n
        A(k + 1:n, j) = A(k + 1:n, j) - A(k + 1:n, k) * A(j, k);
    end
end
fprintf('CPU time used: %f\n', cputime - t);
fprintf('Elapsed time used: %f\n\n', toc);

% Three-for version
fprintf('Three-for version\n');
t = cputime; tic;
A = M;
for k = 1:n
    A(k:n, k) = A(k:n, k) / sqrt(A(k, k));
    A(k, k + 1:n) = 0;
    for j = k + 1:n
        for i = k + 1:n
            A(i, j) = A(i, j) - A(i, k) * A(j, k);
        end
    end
end
fprintf('CPU time used: %f\n', cputime - t);
fprintf('Elapsed time used: %f\n\n', toc);

% Matlab version
fprintf('Matlab version\n');
t = cputime; tic;
A = chol(M, 'lower');
fprintf('CPU time used: %f\n', cputime - t);
fprintf('Elapsed time used: %f\n\n', toc);
```

```
>> main()
One-for version
CPU time used: 1.154407
Elapsed time used: 0.564912

Two-for version
CPU time used: 1.170007
Elapsed time used: 1.044152

Three-for version
CPU time used: 1.294808
Elapsed time used: 1.291551

Matlab version
CPU time used: 0.000000
Elapsed time used: 0.005667

fx >>
```