Λύση 4ης άσκησης
(cache optimizations)

Αρχική Έκδοση

- [DataL1]: size=32*1024 , assoc=4, bsize=64
- [L2Cache]: size=256*1024, assoc=8, bsize=64
- gcc: -O2
- N=256

L1 missrate: 21%

L2 missrate: 12.6%

Instructions: 237M

Cycles: 715M

```
int main(int argc, char **argv)
{
    int i,j,k,N=atoi(argv[1]);
    float **A,**B,**C;
    A=(float**)malloc(N*sizeof(float*));
    for(i=0; i<N; i++)
        A[i]=(float*)malloc(N*sizeof(float));
    B=(float**)malloc(N*sizeof(float*));
    for(i=0; i<N; i++)
        B[i]=(float*)malloc(N*sizeof(float));
    C=(float**)malloc(N*sizeof(float*));
    for(i=0; i<N; i++)
        C[i]=(float*)malloc(N*sizeof(float));

    init_matrix(A,N); init_matrix(B,N);
    init_matrix(C,N);

    sesc_simulation_mark();
    for(i=0; i<N; i++) {
        for(j=0; j<N; j++)
            for(k=0; k<N; k++)
                C[i][j] += A[i][k]*B[k][j];
    }
    sesc_simulation_mark();

    return 0;
}
```

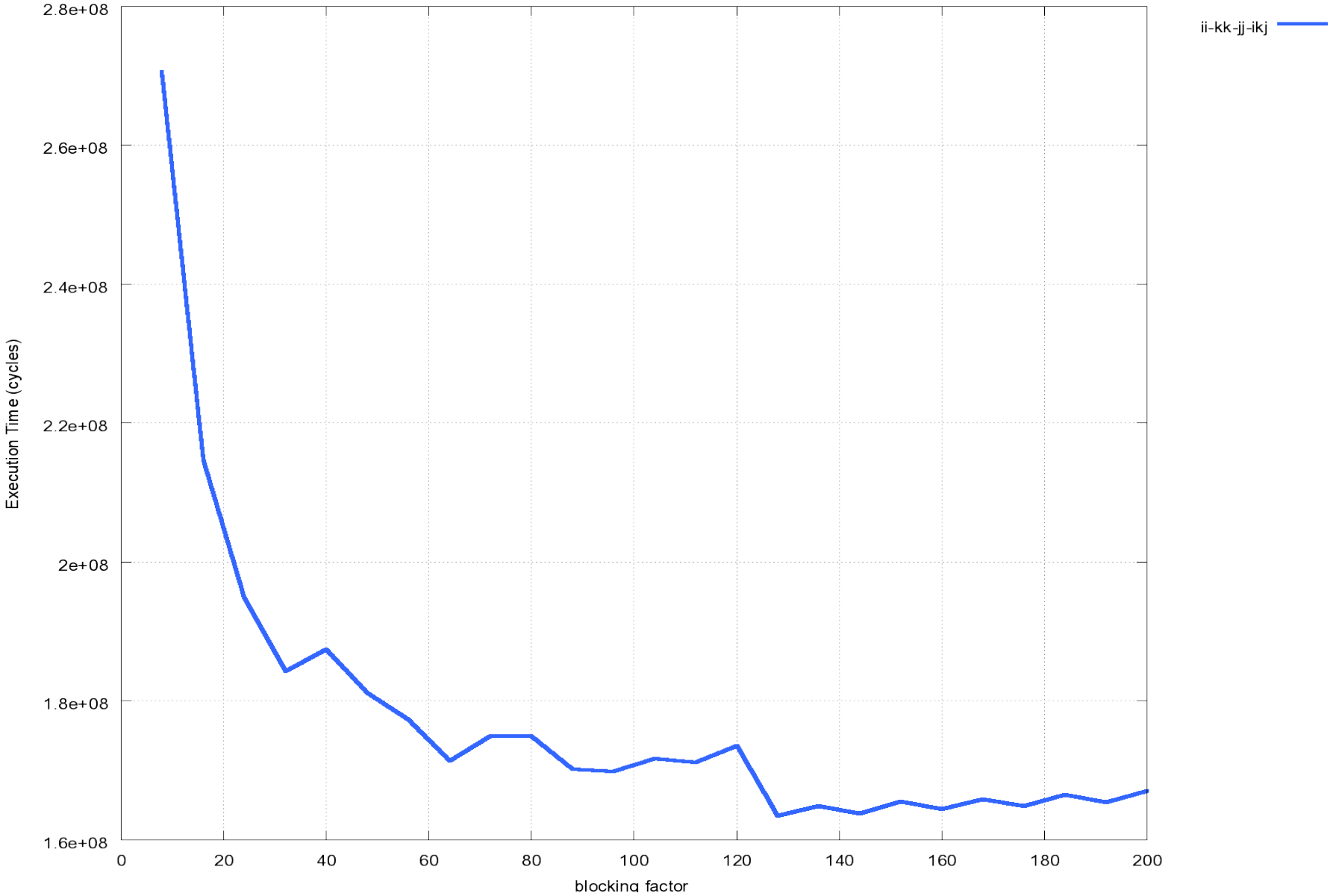
Loop interchange

Αναδιάταξη	L1D miss rate	L2 miss rate	(millions of) Cycles	Speedup
ijk	21.00%	12.26%	715	1
ikj	1.60%	17.38%	194	3,69
jik	21.89%	14.93%	739	0,97
jki	24.10%	18.30%	1013	0,71
kij	1.69%	25.72%	240	2,98
kji	24.04%	17.21%	989	0,72

Cache blocking

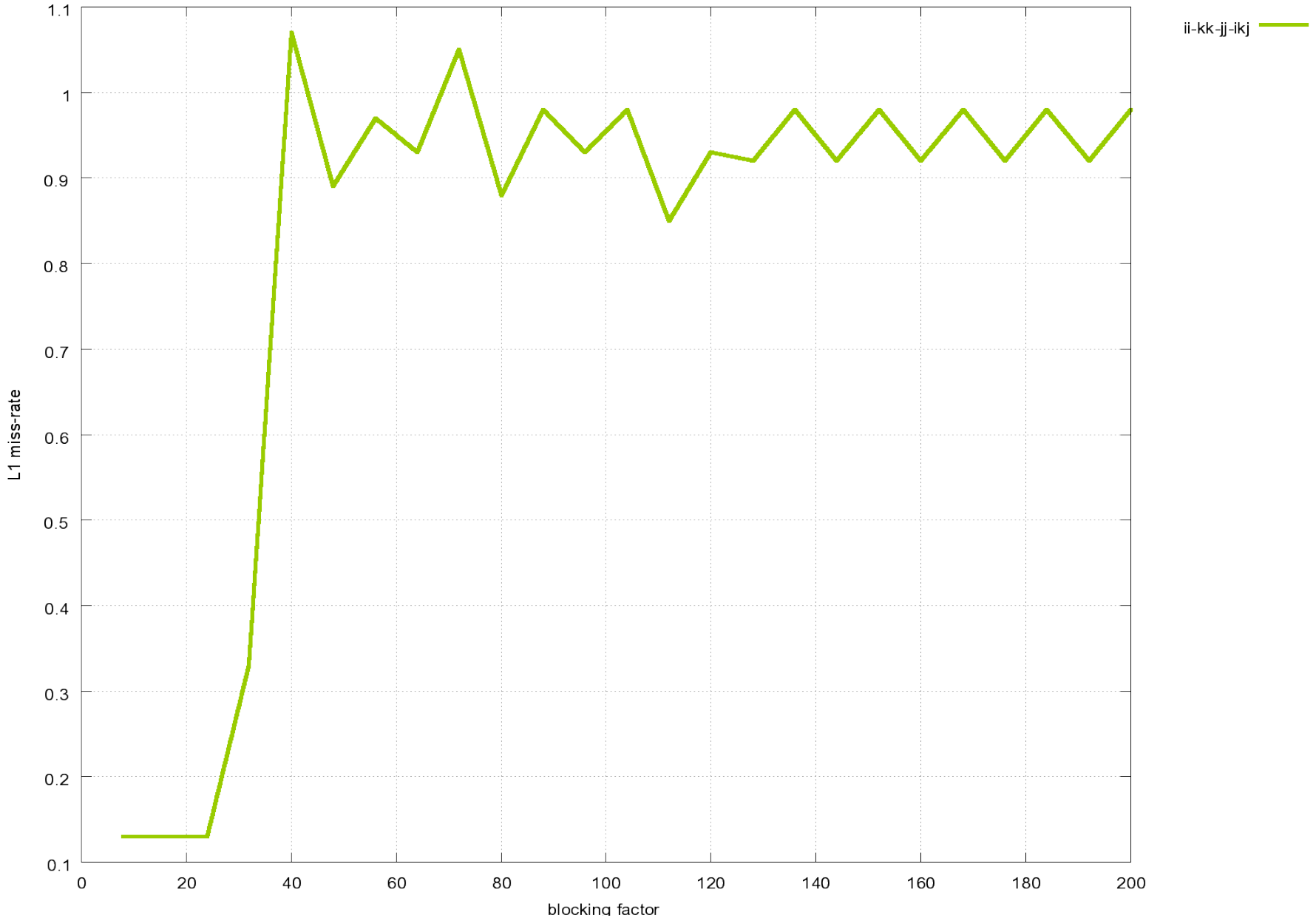
```
for(ii=0; ii<N; ii+=bs)
    for(kk=0; kk<N; kk+=bs)
        for(jj=0; jj<N; jj+=bs)
            for(i=ii; i<min(N,ii+bs); i++)
                for(k=kk; k<min(N,kk+bs); k++)
                    for(j=jj; j<min(N,jj+bs); j++)
                        C[i][j] += A[i][k]*B[k][j];
```

Cache blocking

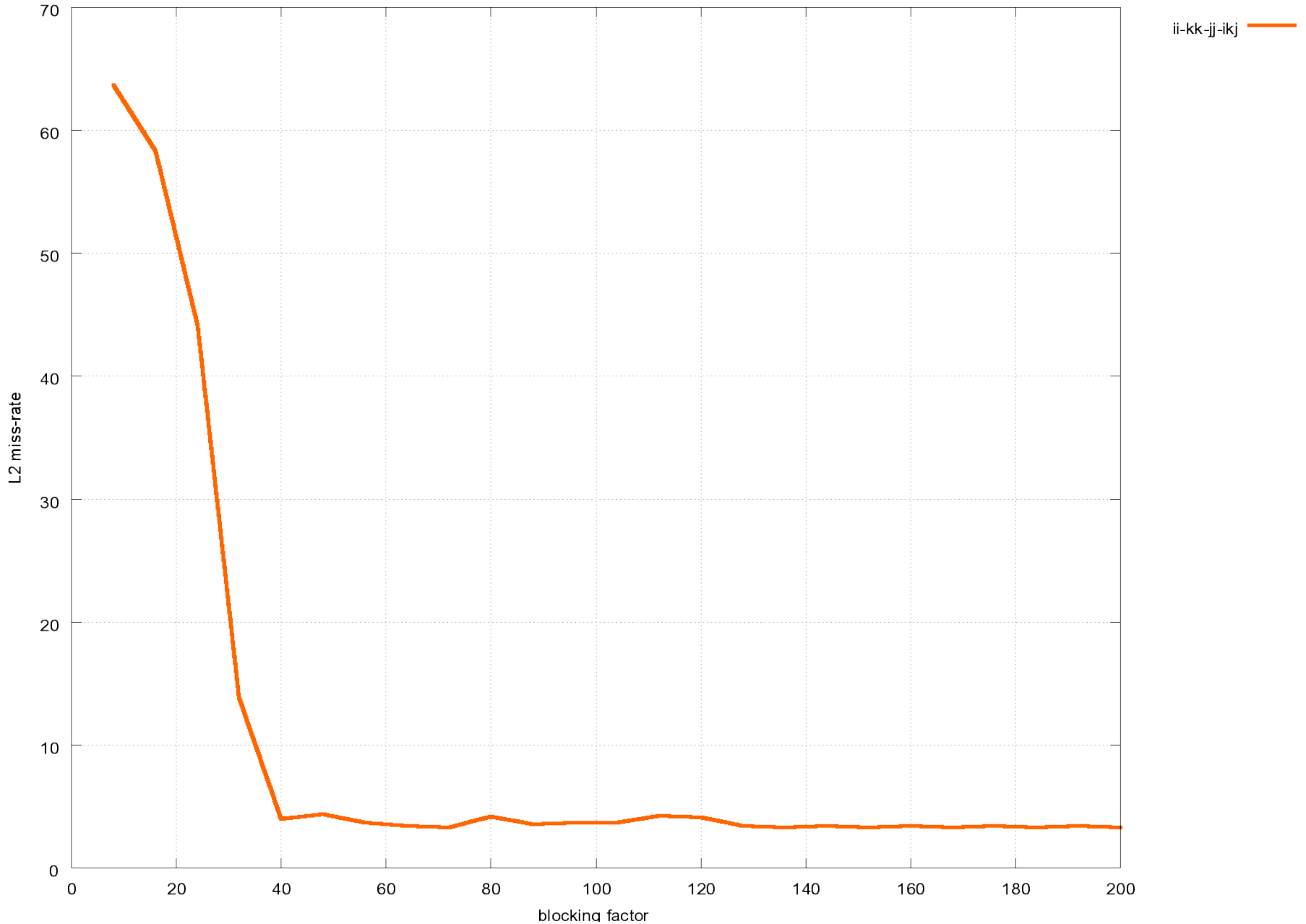


- για bs=128 έχω τους λιγότερους κύκλους (163M), speedup=4.38

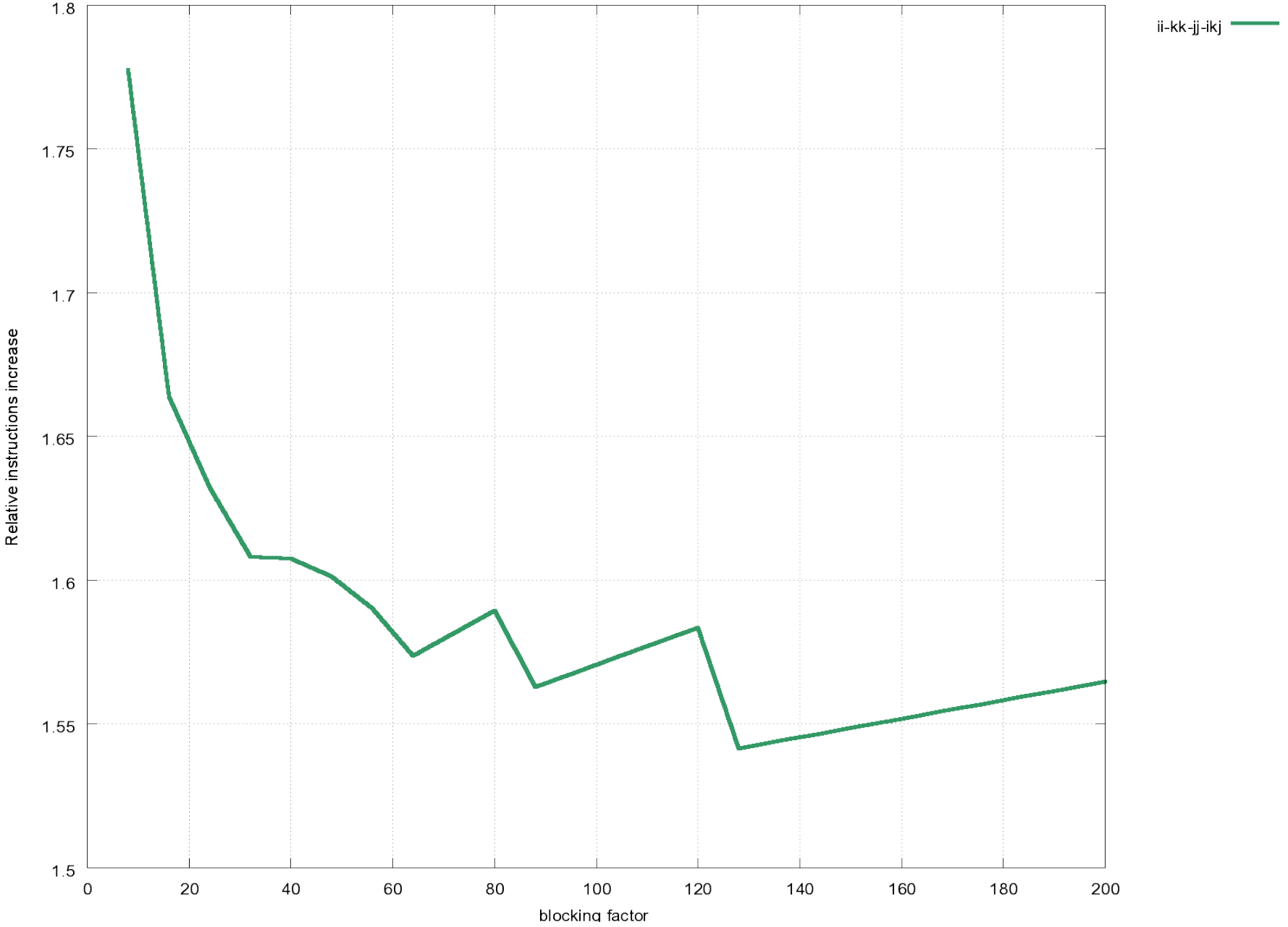
Cache blocking



Cache blocking



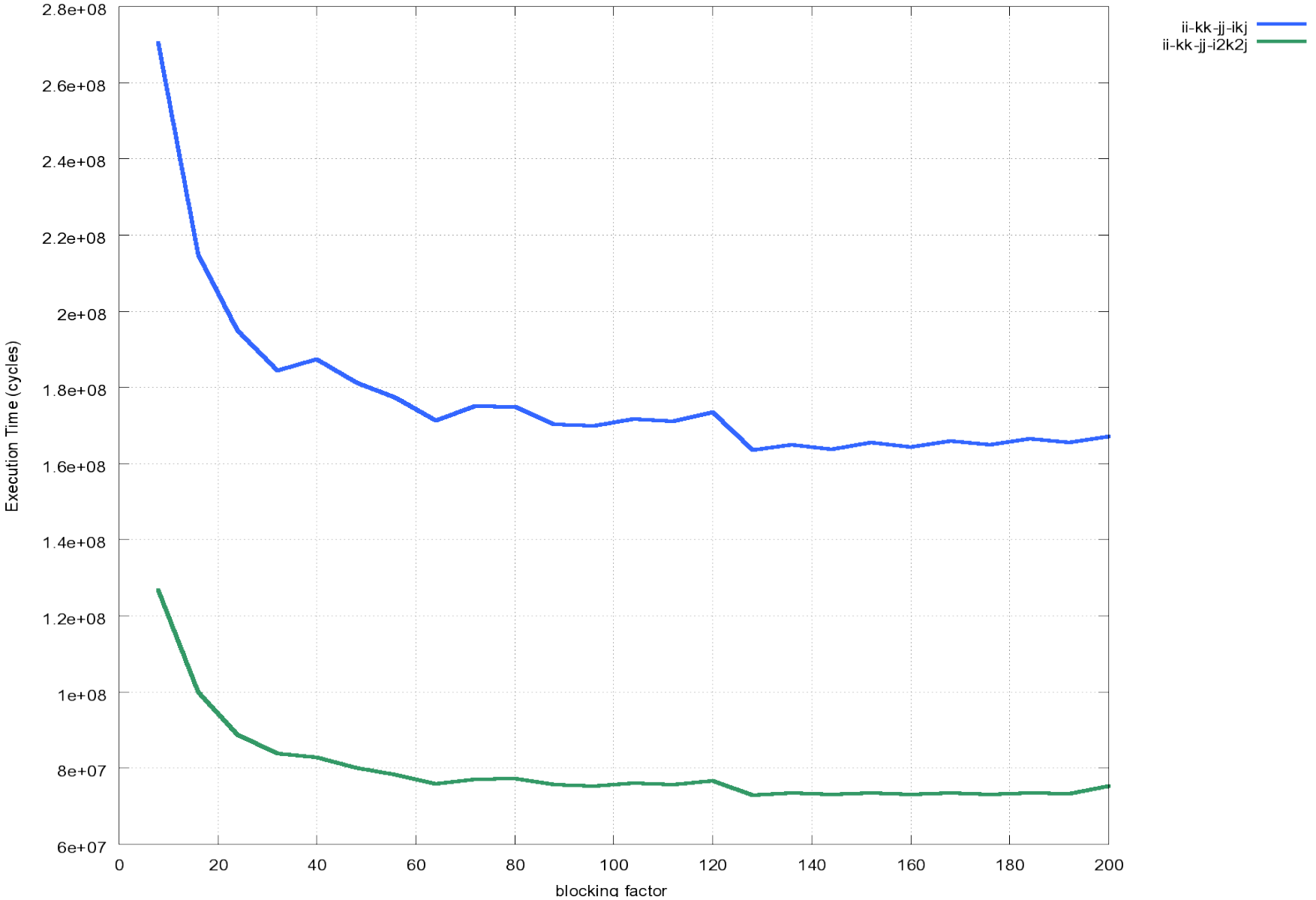
Cache blocking



Register blocking

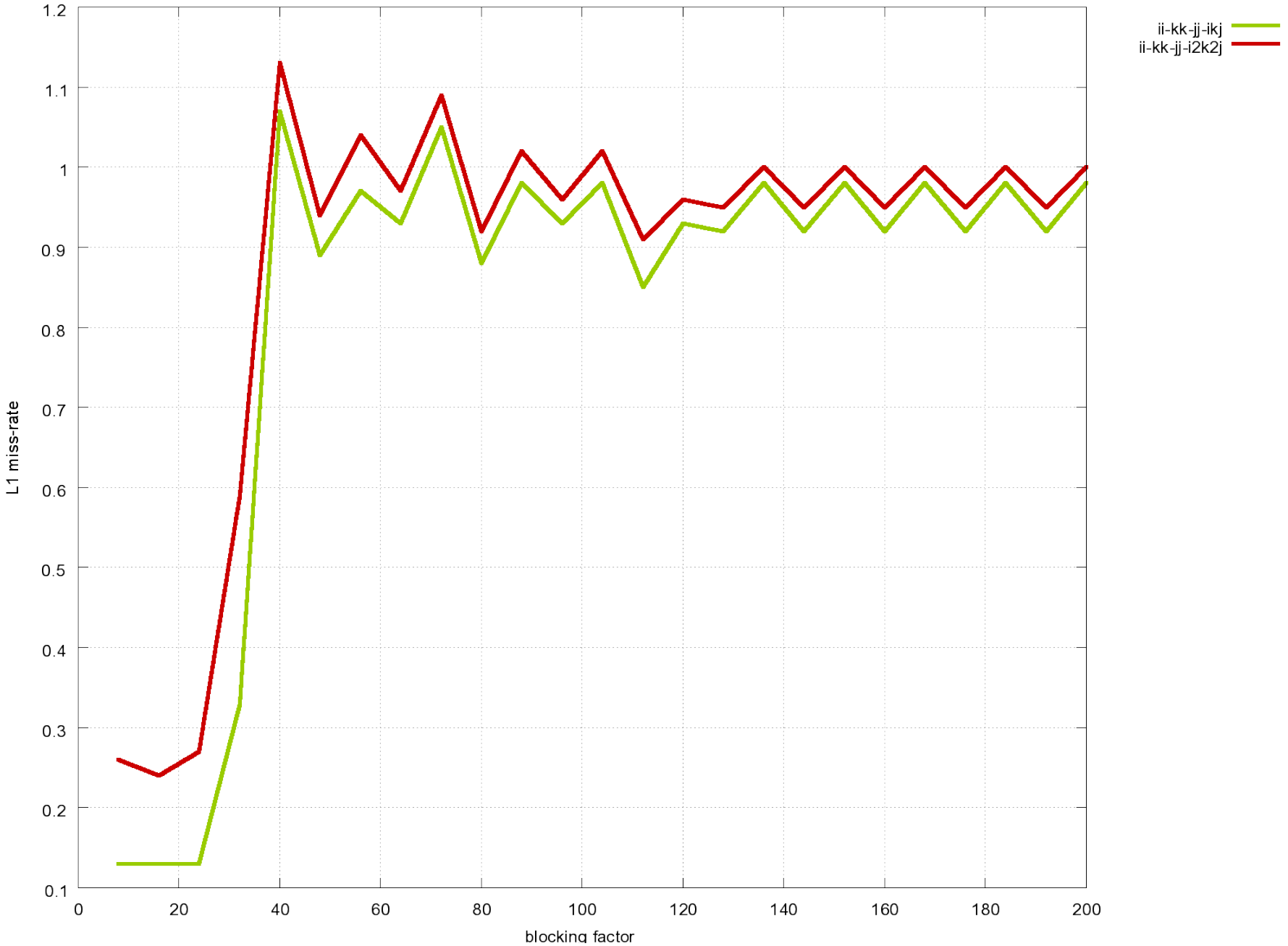
```
...
register float t1,t2,t3,t4;
...
for(ii=0; ii<N; ii+=bs)
    for(kk=0; kk<N; kk+=bs)
        for(jj=0; jj<N; jj+=bs)

            for(i=ii; i<min(N,ii+bs); i+=2) {
                for(k=kk; k<min(N,kk+bs); k+=2) {
                    t1 = A[i][k];
                    t2 = A[i][k+1];
                    t3 = A[i+1][k];
                    t4 = A[i+1][k+1];

                    for(j=jj; j<min(N,jj+bs); j++) {
                        C[i][j] += t1*B[k][j];
                        C[i][j] += t2*B[k+1][j];
                        C[i+1][j] += t3*B[k][j];
                        C[i+1][j] += t4*B[k+1][j];
                    }
                }
            }
```

Register blocking

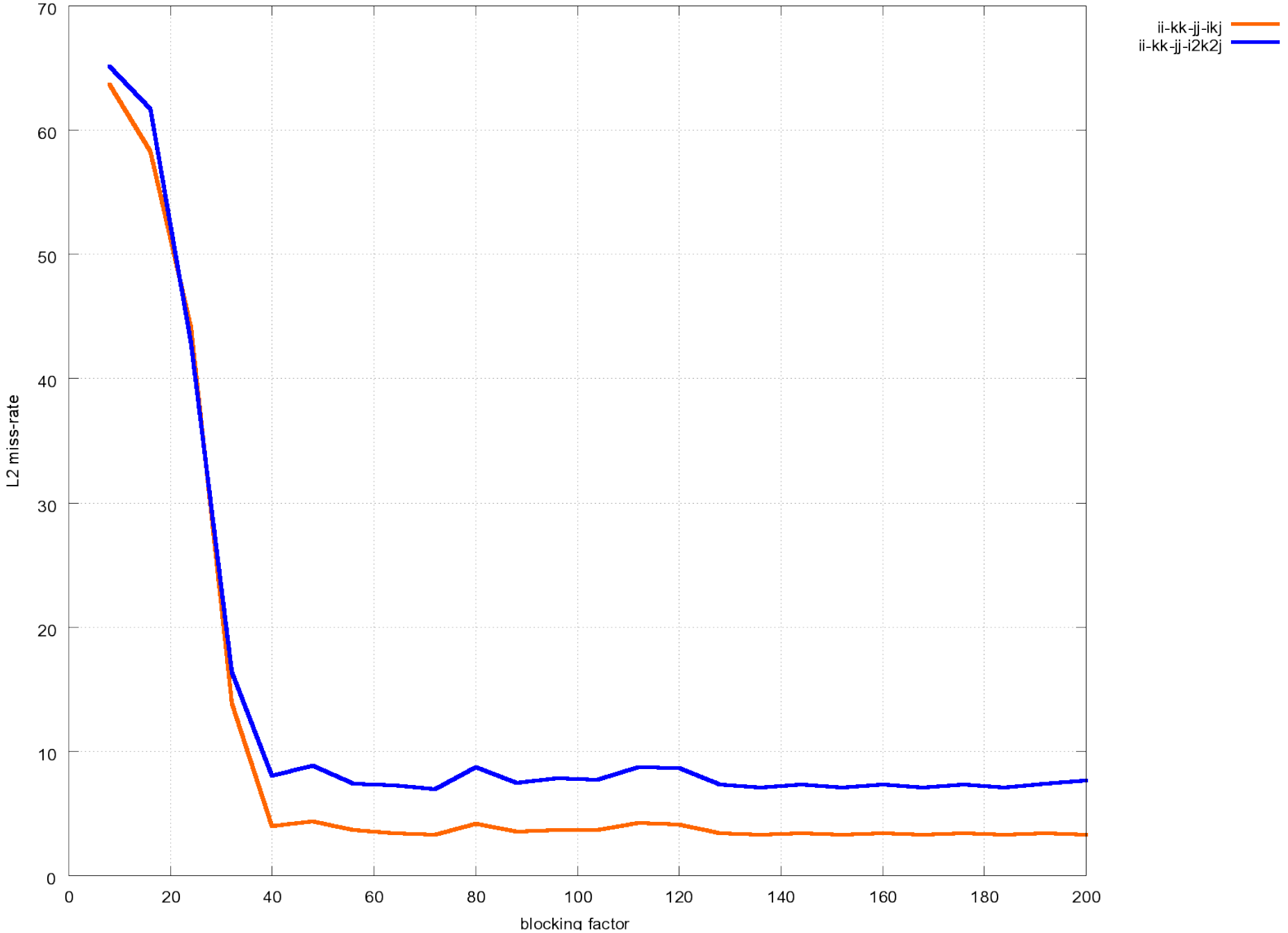


- για bs=128 έχω τους λιγότερους κύκλους (72M), speedup=9.93

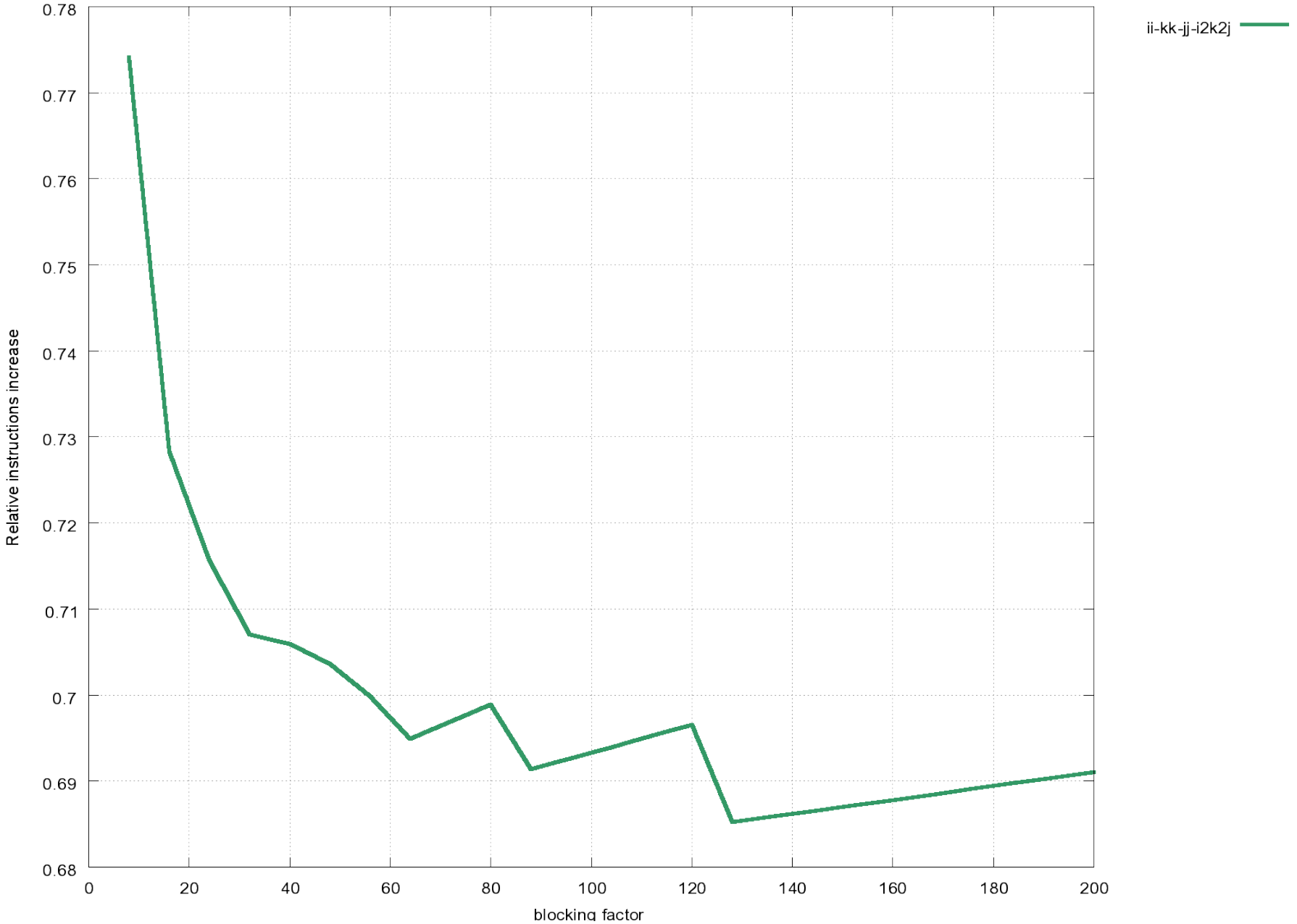
Register blocking



Register blocking



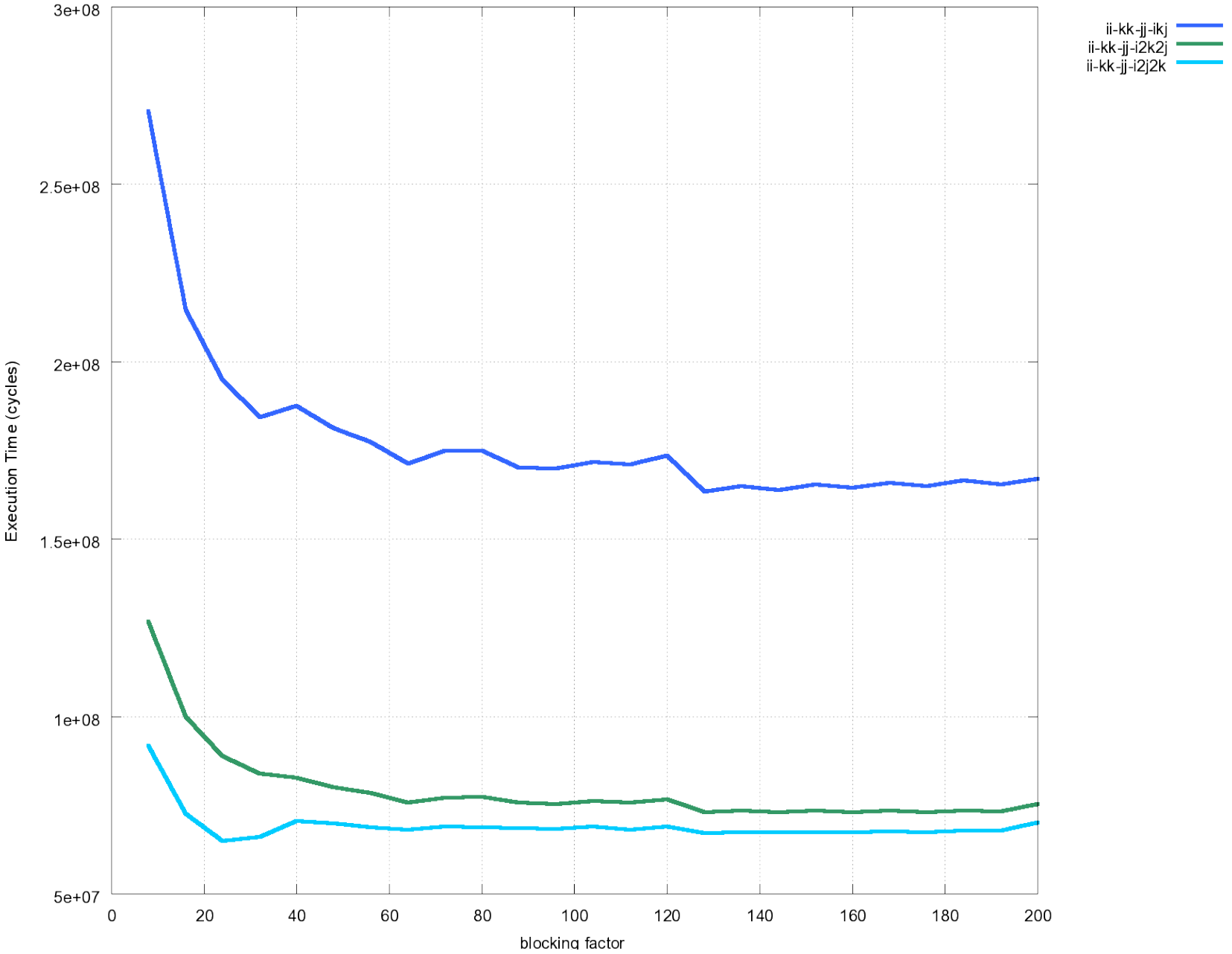
Register blocking



Register blocking (reuse in C)

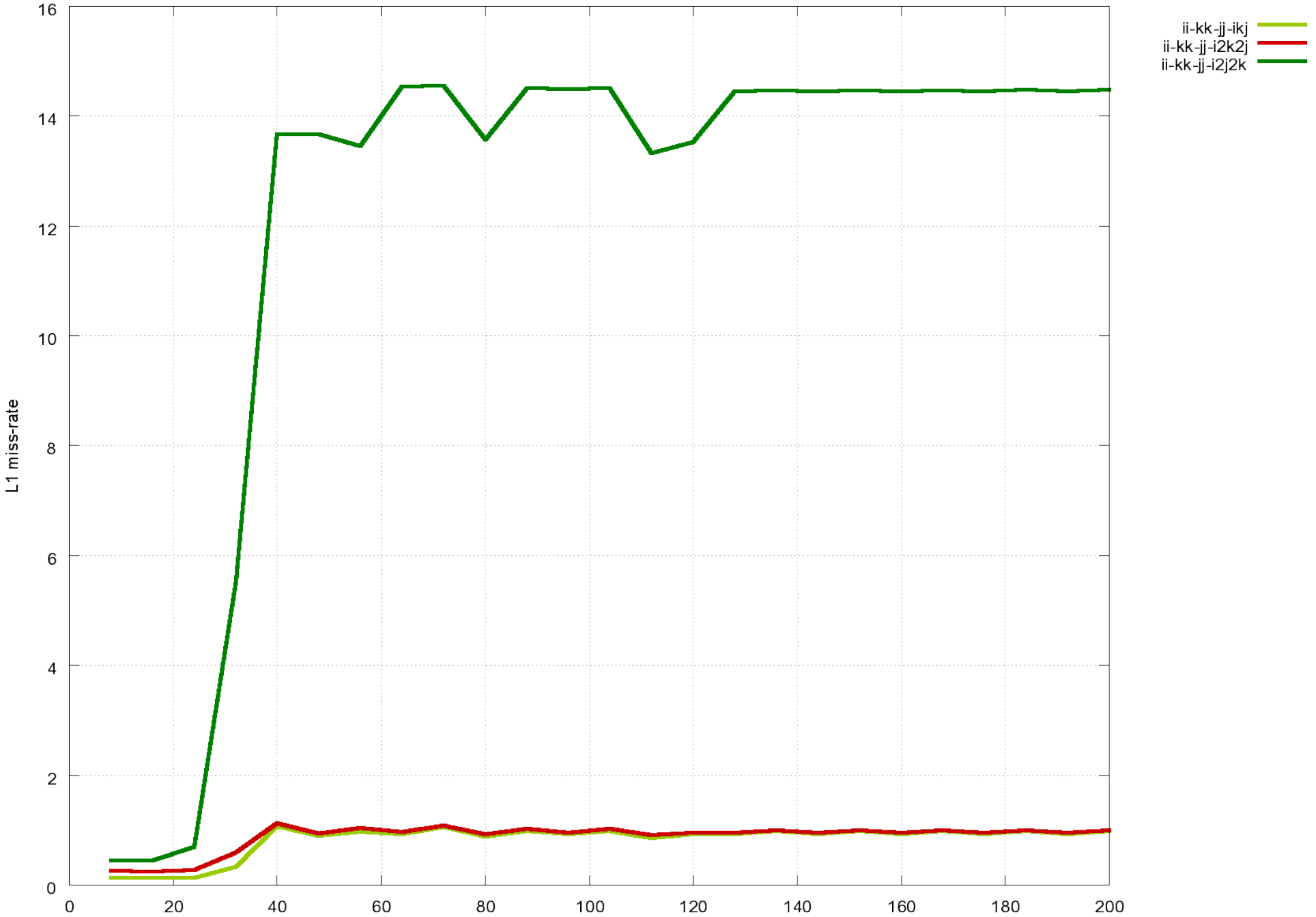
```
...
register float Cij,Cij1,Ci1j,Ci1j1;
...
for(ii=0; ii<N; ii+=bs)
    for(kk=0; kk<N; kk+=bs)
        for(jj=0; jj<N; jj+=bs)

            for(i=ii; i<min(N,ii+bs); i+=2) {
                for(j=jj; j<min(N,jj+bs); j+=2) {
                    Cij = C[i][j];
                    Cij1 = C[i][j+1];
                    Ci1j = C[i+1][j];
                    Ci1j1 = C[i+1][j+1];

                    for(k=kk; k<min(N,kk+bs); k++) {
                        Cij += A[i][k]*B[k][j];
                        Cij1 += A[i][k]*B[k][j+1];
                        Ci1j += A[i+1][k]*B[k][j];
                        Ci1j1 += A[i+1][k]*B[k][j+1];
                    }
                    C[i][j] = Cij;
                    C[i][j+1] = Cij1;
                    C[i+1][j] = Ci1j;
                    C[i+1][j+1] = Ci1j1;
                }
            }
```

Register blocking (reuse in C)

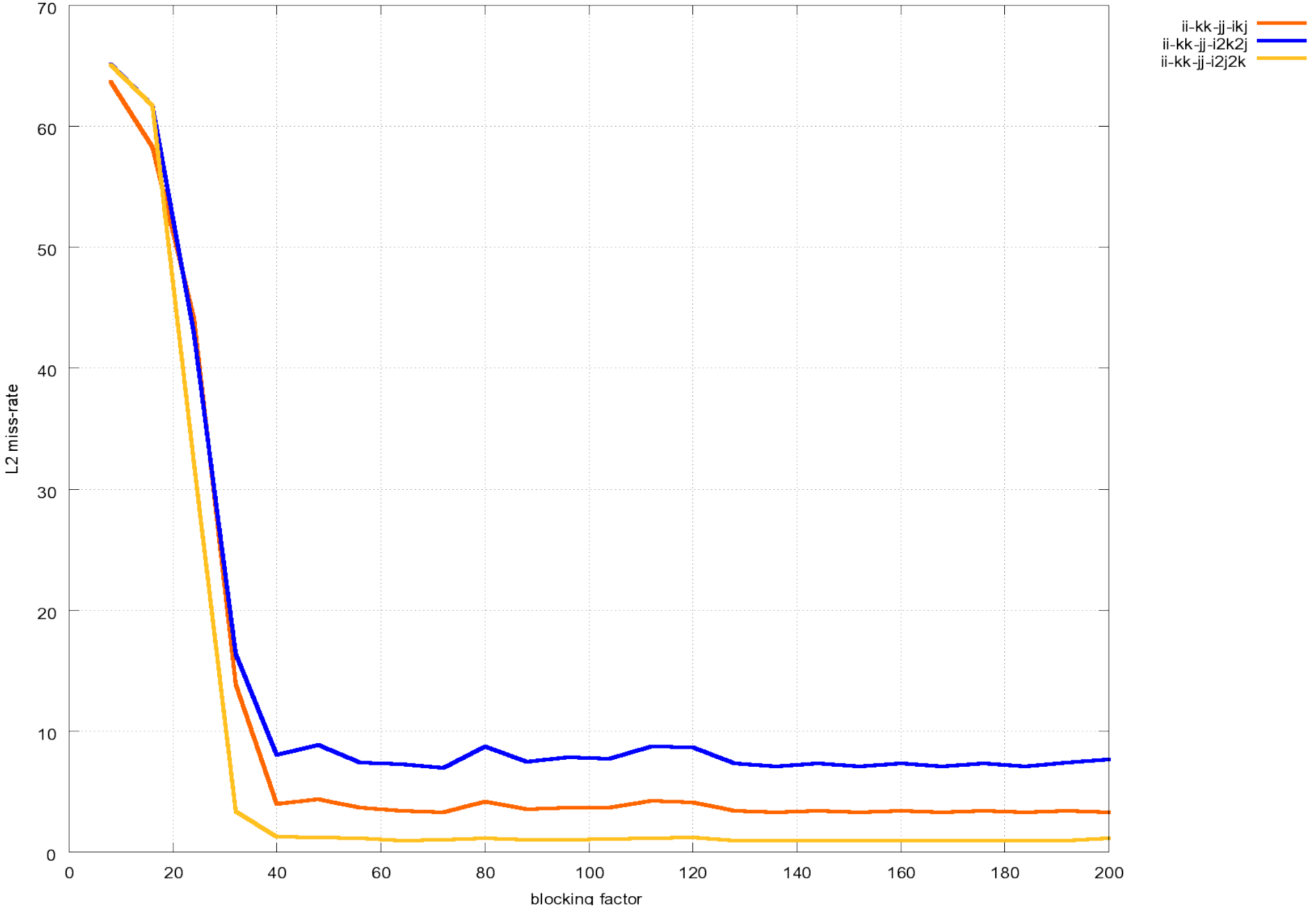


- για bs=24 έχω τους λιγότερους κύκλους (64M), speedup=11.17

Register blocking (reuse in C)



Register blocking (reuse in C)



Συνολική εκτίμηση

Optimization	Speedup
Baseline	1
Loop interchange	3.69
Cache blocking	4.38
Register blocking	9.93
Register blocking (reuse in C)	11.17