

## Procedura testowa mierzenia wydajności

Procedura testowa mierzenia wydajności procesorów zestawów komputerowych (kod źródłowy testu zawarty został w pkt. 4)

1. Uruchomienie zestawu z CD-ROM'u startowego  
CD-ROM startowy będzie zawiera środowisko testowe dostępne na stronie [www.knoppix.org](http://www.knoppix.org) w konfiguracji:  
Wersja dystrybucyjna: KNOPPIX\_V3.4-2004-05-17-EN.iso dostępna w sieci internet  
System operacyjny: Knoppix 3.4  
Jądro: 2.4.26 -dostępne z w/w systemem  
Kompilator: gcc 3.3.3 (Debian 20040429) -dostępne z w/w systemem
2. Kompilacja kodu źródłowego testu  
Do testów wykorzystany kod jest autorstwa Dr. of Eng. Ryutaro Himeno publikowany na stronie [http://accr.riken.jp/E/HPC\\_e/HimenoBMT\\_e/index\\_e.html](http://accr.riken.jp/E/HPC_e/HimenoBMT_e/index_e.html) oraz dostępny w CDO. Kod źródłowy znajduje się w pkt 4.

Uruchomienie systemu:

po znaku zachęty **BOOT**: wpisujemy polecenie **knoppix 2**

Kompilacja:

**gcc -O3 -DSMALL himenobmt.c -o himenobmt.o**

3. 25 krotny pomiar wydajności. Wynikiem pomiaru jest ilość punktów uzyskanych oznaczonych jako „MFLOPS measured”.  
Pomiaru wydajności dokonujemy za pomocą poniższego skryptu:

```
# tester.sh
# !/bin/bash
licz=25
while (($licz>0))
do
    echo "-----" >> tester_out.txt
    echo "POMIAR $licz" >> tester_out.txt
    ./himenobmt.o >> tester_out.txt
    licz=$((licz - 1))
done
```

Skopiowanie wyników 25 pomiarów na dyskietkę odbywa się z poziomu root wykonując kolejno polecenia:

```
mount /dev/fd0 /mnt/floppy
cd /mnt/floppy
gcc -O3 -DSMALL himenobmt.c -o himenobmt.o
./tester.sh
```

4. Kod źródłowy himenoBMT.c

Wykorzystany kod jest autorstwa Dr. of Eng. Ryutaro Himeno. Został on udostępniony jako „open source” i opublikowany w Internecie. Adres strony:

[http://accr.riken.jp/E/HPC\\_e/HimenoBMT\\_e/index\\_e.html](http://accr.riken.jp/E/HPC_e/HimenoBMT_e/index_e.html)

```
/*****
*
```

*This benchmark test program is measuring a cpu performance of floating point operation and memory access speed.*

*Modification needed for testing target computer!!  
Please adjust parameter : nn to take one minute to execute all calculation. Original parameter set is for PC with 200 MHz MMX PENTIUM, whose score using this benchmark test is about 32.3 MFLOPS.*

*If you have any question, please ask me via email.  
written by Ryutaro HIMENO, October 3, 1998.  
Version 2.0*

-----  
*Ryutaro Himeno, Dr. of Eng.  
Head of Computer Information Center,  
The Institute of Physical and Chemical Research  
(RIKEN)  
Email : himeno@postman.riken.go.jp*

-----  
*You can adjust the size of this benchmark code to fit your target computer. In that case, please chose following sets of (mimax,mjmax,mkmax):  
small : 129,65,65  
midium: 257,129,129  
large : 513,257,257  
ext.large: 1025,513,513*

*This program is to measure a computer performance in MFLOPS by using a kernel which appears in a linear solver of pressure Poisson included in an incompressible Navier-Stokes solver. A point-Jacobi method is employed in this solver.*

-----  
*Finite-difference method, curvilinear coordinate system  
Vectorizable and parallelizable on each grid point  
No. of grid points : imax x jmax x kmax including boundaries*

-----  
*A,B,C:coefficient matrix, wrk1: source term of Poisson equation  
wrk2 : working area, OMEGA : relaxation parameter  
BND:control variable for boundaries and objects ( = 0 or 1)  
P: pressure*

-----  
*"use portlib" statement on the next line is for Visual fortran to use UNIX libraries. Please remove it if your system is UNIX.*

-----  
*use portlib*

Version 0.2

```
*****
*/

#include <stdio.h>

#ifdef SMALL
#define MIMAX      129
#define MJMAX      65
#define MKMAX      65
#endif

#ifdef MIDDLE
#define MIMAX      257
#define MJMAX      129
#define MKMAX      129
#endif

#ifdef LARGE
#define MIMAX      513
#define MJMAX      257
#define MKMAX      257
#endif

#define NN          200

double second();
float jacobi(int);
void initmt();

static float p[MIMAX][MJMAX][MKMAX];
static float a[MIMAX][MJMAX][MKMAX][4],
             b[MIMAX][MJMAX][MKMAX][3],
             c[MIMAX][MJMAX][MKMAX][3];
static float bnd[MIMAX][MJMAX][MKMAX];
static float wrk1[MIMAX][MJMAX][MKMAX],
             wrk2[MIMAX][MJMAX][MKMAX];

static int imax, jmax, kmax;
static float omega;

int
main()
{
    int i, j, k;
    float gosa;
    double cpu0, cpu1, nflop, xmflops2, score;

    omega = 0.8;
    imax = MIMAX-1;
    jmax = MJMAX-1;
    kmax = MKMAX-1;

    /*
     *   Initializing matrixes
     */
}
```

```

initmt();

printf("mimax = %d mjmax = %d mkmax = %d\n",MIMAX,
MJMAX, MKMAX);
printf("imax = %d jmax = %d kmax
=%d\n",imax,jmax,kmax);

/*
 * Start measuring
 */
cpu0 = second();

/*
 * Jacobi iteration
 */

gosa = jacobi(NN);

cpu1 = second();

nflop = (kmax-2)*(jmax-2)*(imax-2)*34;

if(cpu1 != 0.0)
    xmflops2 = nflop/cpu1*1.0e-6*(float)NN;

score = xmflops2/32.27;

printf("cpu : %f sec.\n", cpu1);
printf("Loop executed for %d times\n",NN);
printf("Gosa : %e \n",gosa);
printf("MFLOPS measured : %f\n",xmflops2);
printf("Score based on MMX Pentium 200MHz :
%f\n",score);

return (0);
}

void
initmt()
{
    int i,j,k;

for(i=0 ; i<imax ; ++i)
for(j=0 ; j<jmax ; ++j)
for(k=0 ; k<kmax ; ++k){
    a[i][j][k][0]=0.0;
    a[i][j][k][1]=0.0;
    a[i][j][k][2]=0.0;
    a[i][j][k][3]=0.0;
    b[i][j][k][0]=0.0;
    b[i][j][k][1]=0.0;
    b[i][j][k][2]=0.0;
    c[i][j][k][0]=0.0;
    c[i][j][k][1]=0.0;
    c[i][j][k][2]=0.0;
    p[i][j][k]=0.0;
    wrk1[i][j][k]=0.0;
}
}

```

```

        bnd[i][j][k]=0.0;
    }

    for(i=0 ; i<imax ; ++i)
        for(j=0 ; j<jmax ; ++j)
            for(k=0 ; k<kmax ; ++k){
                a[i][j][k][0]=1.0;
                a[i][j][k][1]=1.0;
                a[i][j][k][2]=1.0;
                a[i][j][k][3]=1.0/6.0;
                b[i][j][k][0]=0.0;
                b[i][j][k][1]=0.0;
                b[i][j][k][2]=0.0;
                c[i][j][k][0]=1.0;
                c[i][j][k][1]=1.0;
                c[i][j][k][2]=1.0;
                p[i][j][k]=(float)(k*k)/(float)((kmax-1)*(kmax-1));
                wrk1[i][j][k]=0.0;
                bnd[i][j][k]=1.0;
            }
    }

```

```

float
jacobi(int nn)
{
    int i,j,k,n;
    float gosa, s0, ss;

    for(n=0;n<nn;++n){
        gosa = 0.0;

        for(i=1 ; i<imax-1 ; ++i)
            for(j=1 ; j<jmax-1 ; ++j)
                for(k=1 ; k<kmax-1 ; ++k){
                    s0 = a[i][j][k][0] * p[i+1][j][k]
                        + a[i][j][k][1] * p[i][j+1][k]
                        + a[i][j][k][2] * p[i][j][k+1]
                        + b[i][j][k][0] * ( p[i+1][j+1][k] - p[i+1][j-1][k]
                            - p[i-1][j+1][k] + p[i-1][j-1][k] )
                        + b[i][j][k][1] * ( p[i][j+1][k+1] - p[i][j-1][k+1]
                            - p[i][j+1][k-1] + p[i][j-1][k-1] )
                        + b[i][j][k][2] * ( p[i+1][j][k+1] - p[i-1][j][k+1]
                            - p[i+1][j][k-1] + p[i-1][j][k-1] )
                        + c[i][j][k][0] * p[i-1][j][k]
                        + c[i][j][k][1] * p[i][j-1][k]
                        + c[i][j][k][2] * p[i][j][k-1]
                        + wrk1[i][j][k];

                    ss = ( s0 * a[i][j][k][3] - p[i][j][k] ) * bnd[i][j][k];

                    gosa = gosa + ss*ss;

                    wrk2[i][j][k] = p[i][j][k] + omega * ss;
                }

        for(i=1 ; i<imax-1 ; ++i)
            for(j=1 ; j<jmax-1 ; ++j)

```

```

        for(k=1 ; k<kmax-1 ; ++k)
            p[i][j][k] = wrk2[i][j][k];
    } /* end n loop */

    return(gosa);
}

double
second()
{
#include <sys/time.h>

    struct timeval tm;
    double t ;

    static int base_sec = 0,base_usec = 0;

    gettimeofday(&tm, NULL);

    if(base_sec == 0 && base_usec == 0)
    {
        base_sec = tm.tv_sec;
        base_usec = tm.tv_usec;
        t = 0.0;
    } else {
        t = (double) (tm.tv_sec-base_sec) +
            ((double) (tm.tv_usec-base_usec))/1.0e6 ;
    }

    return t ;
}

```