Exercise 1

Information theory and computation

Davide Maniscalco

1. Setup

The working directory is created with the command `mkdir`, while the first program was written in class. The connection to the remote server is made with the command `ssh maniscad@spiro.fisica.unipd.it`, and inserting afterwards the same credentials of the lab. Instead, to push a file to remote in order to work with it, the command to be used is `scp file.f maniscad@spiro.fisica.unipd.it:info_th/es1` (where the directories 'info_th' and 'es1' were created in advance).

2. Number precision

The code is the following:

```
      SUBROUTINE POINT2A
      INTEGER*4 x,y
      INTEGER*2 q,w
      x=1
      y=2000000
      q=1
      w=2000000
      print*,'Sum with integer*4: ' , x+y
      print*,'Sum with integer*2: ', q+w
      END SUBROUTINE POINT2A

    SUBROUTINE POINT2B
      REAL*8 x,y
      REAL*4 q,w
      x=4*ATAN(1.0)*1E32
      y=sqrt(2.0)*1E21
      q=4*ATAN(1.0)*1E32
      w=sqrt(2.0)*1E21
      print*,'Sum with double precision: ',x+y
      print*,'Sum with single precision: ',q+w
      print*,'(pi)E32 in single precision: ',q
      END SUBROUTINE POINT2B

      PROGRAM SHEET1PART1
      IMPLICIT NONE
      call POINT2A()
      call POINT2B()
      STOP
      END
```

It produces the output:

```
Sum with integer*4: 2000001
Sum with integer*2: -31615
Sum with double precision: 3.1415927798296973E+032
Sum with single precision: 3.14159278E+32
(pi)E32 in single precision: 3.14159278E+32
```

The 4-byte integer allocation allows to obtain the correct result, while the 2-byte doesn't. Let's remind that the range of integer numbers represantable with n bits is $[-2^{n-1}, 2^{n-1} - 1]$; in our case we have 16 bits (2 bytes), and therefore the maximum representable number is 32767. Because of the lack of space, the program truncates the bits of the result, leading to a completely wrong number in the decimal representation.

The second sum is instead possibile with both single and double precision. Here the problem is that the single precision is not enough to show the difference between $\pi \cdot 10^{32} + \sqrt{2} \cdot 10^{21}$ and $\pi \cdot 10^{32}$.

3. Test performance

The code is the following:

```
      PROGRAM SHEET1PART2
      IMPLICIT NONE
      DOUBLE PRECISION,DIMENSION(2000,2000) ::mat1
      DOUBLE PRECISION, DIMENSION (2000,2000) :: mat2
      DOUBLE PRECISION,DIMENSION(2000,2000) ::prodmat1,
   prodmat2,prodmat3

      INTEGER ii,jj,kk,nn,mm,ll
      REAL T1,T2,T3,T4,T5,T6

! Product between matrices of dimensions: (nn x mm)(mm x ll)=(
   nn x ll)
      nn = 2000
      mm = 2000
      ll = 2000

! definition of the first matrix
       DO ii=1,nn
         DO jj=1,mm
            mat1(ii,jj)=ii
         ENDDO
      ENDDO

! definition of the second matrix
      DO ii=1,mm
         DO jj=1,ll
            mat2(ii,jj)=jj
         ENDDO
      ENDDO

! By column loop
      CALL CPU_TIME(T3)
      DO kk=1,mm
         DO jj=1,ll
            DO ii=1,nn
              prodmat1(ii,jj)=prodmat1(ii,jj)+mat1(ii,kk)*mat2
   (kk,jj)
            ENDDO
         ENDDO
      ENDDO
```

```
      CALL CPU_TIME(T4)

! By row loop
      CALL CPU_TIME(T1)
      DO ii=1,nn
         DO jj=1,ll
            DO kk=1,mm
               prodmat2(ii,jj)=prodmat2(ii,jj)+mat1(ii,kk)*
   mat2(kk,jj)
            ENDDO
         ENDDO
      ENDDO
      CALL CPU_TIME(T2)

! Fortran function
      CALL CPU_TIME(T5)
      prodmat3 = MATMUL(mat1,mat2)
      CALL CPU_TIME(T6)

! print of output matrices, uncomment to check correctness of
   the result
!      print*, prodmat1
!      print*, prodmat2
!      print*, prodmat3

    print*, 'Input Matrices dimensions: ',nn,'',mm,';',mm,'',
   ll
      print*, 'By row time: ',T2-T1
      print*, 'By col time: ',T4-T3
      print*, 'Matmul time: ',T6-T5

      STOP
      END
```

Matrix-matrix product for matrices of any dimension was implemented, first filling the final matrix by row, making with the loops the usual row-column product. Afterwards, the filling of the last matrix is made by column, and also the product is made differently, i.e. swapping the first and the last loop, creating a kind of transposition of the first matrix and a column-column product. Finally the predefined `mathmul` function was used. For the time measurements only squared matrices were taken into account just for simplicity; input matrices were defined using two do-loops. Matrices size was manually modified in the code; many changes were made in order to monitor CPU time. Here below the results:

Dimension	10	100	500	1000	2000
By row time (s)	0	$1.6 \cdot 10^{-2}$	2.112	14.648	112.028
By column time (s)	0	$2 \cdot 10^{-2}$	1.508	12.064	96.13
Matmul time(s)	0	0	0.137	1.704	15.24
Compilation time (s)	0.128	0.126	0.129	0.129	0.128

Because of not perfectly known reasons, maybe related to Fortran architecture, the product by column is a bit quicker than the by row one, except at low dimensions, while the

predefined function is much faster. As expected, the matrix dimension doesn't affect the compilation time.

Finally, we want to analyze the impact of the optimized compilation, made with the flag -O. Time measurements were all taken again:

Dimension	10	100	500	1000	2000
By row time (s)	0	$4 \cdot 10^{-3}$	0.464	2.952	22.26
By column time (s)	0	$4 \cdot 10^{-3}$	0.272	2.576	20.688
Matmul time(s)	0	0	0.136	1.696	14.988
Compilation time (s)	0.145	0.149	0.149	0.149	0.151

As expected, compilation times don't change with the dimension of the matrix, but they are a bit higher than the ones taken with the normal compilation. On the other hand, this produces big advantages in the execution time for the hand-defined functions: the execution time becomes smaller of one order of magnitude. The execution time of the `matmul` function is smaller too, but in a very less dramatical way.

Measuruements were taken also for the other compilier optimizations -O1, -O2, -O3; there is not a qualitatively different behaviour for any of them. Results are shown in the following tables.

-O1

Dimension	10	100	500	1000	2000
By row time (s)	0	$4 \cdot 10^{-3}$	0.464	2.960	23.704
By column time (s)	0	$4 \cdot 10^{-3}$	0.276	2.584	20.520
Matmul time(s)	0	0	0.139	1.692	15.216
Compilation time (s)	0.146	0.151	0.151	0.154	0.149

-O2

Dimension	10	100	500	1000	2000
By row time (s)	0	$4 \cdot 10^{-3}$	0.424	2.832	22.148
By column time (s)	0	$4 \cdot 10^{-3}$	0.276	2.568	20.588
Matmul time(s)	0	0	0.136	1.687	15.112
Compilation time (s)	0.154	0.168	0.172	0.173	0.170

-O3

Dimension	10	100	500	1000	2000
By row time (s)	0	$4 \cdot 10^{-3}$	0.452	2.916	22.964
By column time (s)	0	$4 \cdot 10^{-3}$	0.203	2.352	18.788
Matmul time(s)	0	0	0.136	1.679	15.204
Compilation time (s)	0.173	0.176	0.175	0.176	0.176