Exam: Scientific Programming (Wi4260TU)

Monday, 10 April 2017, 13:30 – 16:30

Part I: Theory

(Use of book and the slides from lectures and lab sessions is allowed)

1. (a) Why is 'Compute intensity' (or 'Data locality') important to the performance of a program on today's computer systems?
   (b) Name two techniques (there are many) that can be applied to 'cache optimization'.

2. (a) What is 'loop unrolling'?
   (b) Can the following loop be parallelized efficiently? Please explain.

```
for (i=0; i<N; i++) {
    A[i]=A[i-1]+A[i+1]
}
```

3. Why is software documentation an important part in the software development process? Name 3 or 4 things that should be documented in the header of a (non-trivial) function.

4. Mathematically the variance of a series of n data (numbers) x(i) can be calculated using one of the following formulas:

$$\text{var}(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x(i)-\bar{x})^2, \text{ or } \text{var}(x) = \frac{1}{n-1}\sum_{i=1}^{n}x^2(i) - \frac{1}{n(n-1)}\left(\sum_{i=1}^{n}x(i)\right)^2$$

   where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n}x(i)$ is the mean.

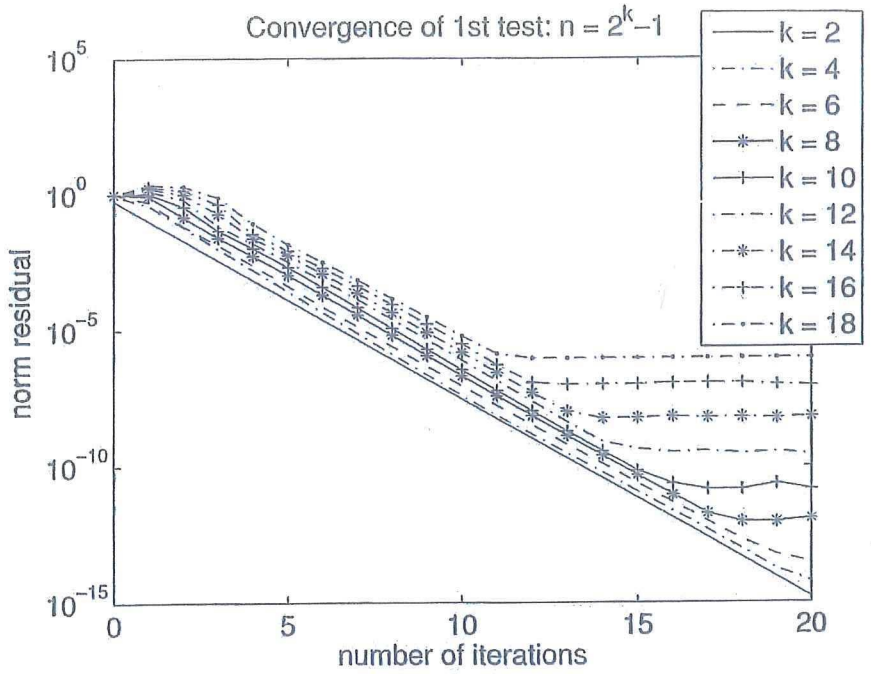
   Given a set of 6 numbers: 5124.75, 5124.48, 5124.90, 5125.31, 5125.05, 5124.98. When single precision floating numbers are used to calculate the variance, we obtain the results 0.07902 and -4.00000 using the first (left) and second (right) formula respectively. Explain how this can happen.

5. (a) A and B N×N matrices defined as a two dimensional array. Consider the following loop in C language

```
for (i=0; i<N; i++)
        for (j=0; j<N; j++)
          A[i][j]=B[i][j]*B[j][i];
```

For large N, what kind of problem in performance loss do you expect?

(b) The following graph shows the plot of the residual norms of the multi-grid iterations for a 1-dimenisional test problem in Chapter 19, why the error (residual) stopped to decrease with increasing number of iterations for large k (i.e., finer grid with more discretization points)?

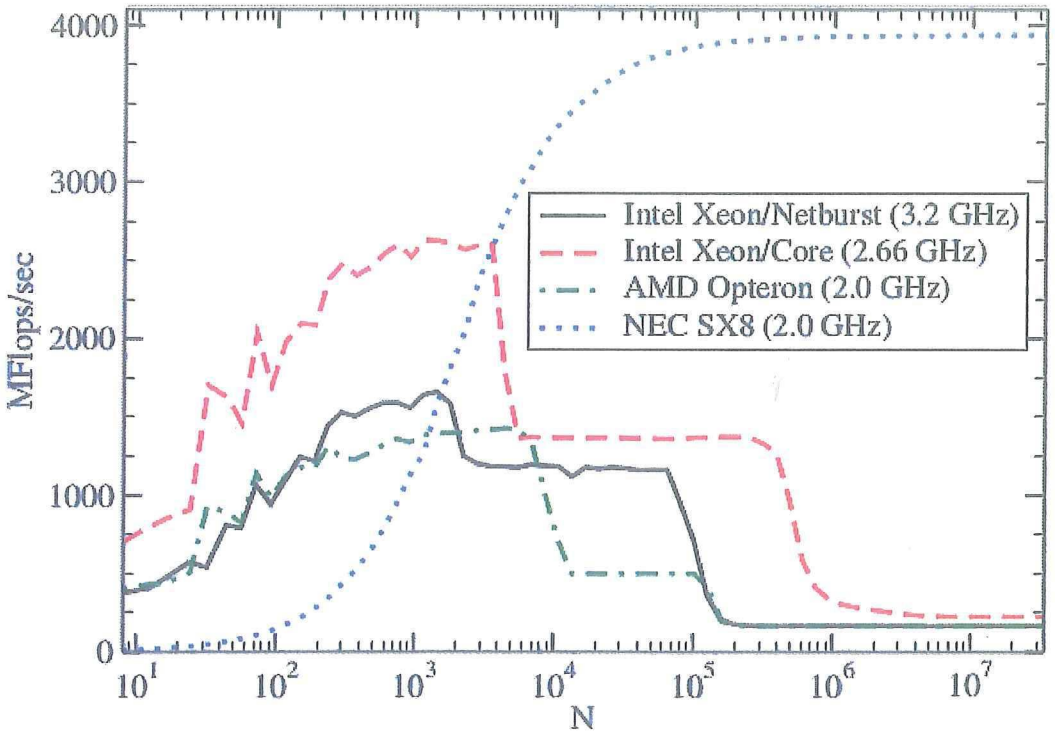


. Plot of residual norms against iteration number for the first test problem.

6. The performance measurements of the following loop (vector triad) is shown in the figure below.

```
DO i=1, N
  A(i) = B(i) + C(i)*D(i)
ENDDO
```

Consider the red-curve, (a) explain why the FLOP rate increases from N=1 to a few thousands, but after that it drops again. (b) What changes in the red-curve can we expect if the size of the L1 (level-1) cache of the Intel Xeon is doubled?



Exam: Scientific Programming April 2017 (Wi4260)

Part II: Lab exercises

Thursday April 10, 2017 ; 13:30-16:30

Some rules

- You may use your text book "Writing Scientific Software".
- You may use a printed version (without notes) of the lab and lecture slides.
- You are not allowed to use the internet, either on mobile, laptop or whatever.
- You are not allowed to use the examples from the lab accounts.

Write your answers on paper and hand it in with your name and study number when finished.

Questions

1. To measure how long a program takes to complete you can either use an **external** command like `time myprogram` or **internal** timer functions like e.g. `tic` in Matlab, `clock_gettime(CLOCK_REALTIME,&t1)` in C or `t1=time()` in Python.

   Discuss their advantages and disadvantages and when you would use either of those methods.

2. a) What is GMP and what can it do that ordinary programs can't?

   b) Why don't we always use GMP if it is so powerful?

   c) If we change the software precision for the mantissa in GMP to 64 bits (the lowest possible value) the result is still slightly better than for a normal C program using standard 64 bit floating point numbers (e.g. on an Intel processor). Why is GMP still better here?

3. If we compute $\frac{1-\cos(x)}{x^2}$ for small x we should get closer to 1/2 if x is divided by 10 after each step:

   $$\lim_{x\to 0} \frac{1-\cos(x)}{x^2} = 1/2$$

   However, if we use a single precision program we find the following result for decreasing values of x:

```
Iteration   0, float result for x=0.10000000 : 0.49958226
Iteration   1, float result for x=0.01000000 : 0.50008297
Iteration   2, float result for x=0.00100000 : 0.47683722
Iteration   3, float result for x=0.00010000 : 0.00000000
Iteration   4, float result for x=0.00001000 : 0.00000000
....
....
Iteration  16, float result for x=0.00000000 : 0.00000000
Iteration  17, float result for x=0.00000000 : 0.00000000
FPE error occurred in float !
```

   a) In the output you can see 2 different types of floating point errors. Where do they occur and what type of errors are they ?

   b) Why do we see an error message for the second one but not for the first one?

   c) How is the error message in this program generated?

4. If we compute the limit from question 3 with an improved algorithm, the result is as follows:

```
Iteration   0, float result for x=0.10000000 : 0.49958351
Iteration   1, float result for x=0.01000000 : 0.49999583
Iteration   2, float result for x=0.00100000 : 0.50000000
Iteration   3, float result for x=0.00010000 : 0.50000000
....
....
Iteration  20, float result for x=0.00000000 : 0.50000000
Iteration  21, float result for x=0.00000000 : 0.50000000
Iteration  22, float result for x=0.00000000 : -nan
```

a) Rewrite the problematic formula from question 3 to an improved one that produces the result from above (there are several solutions possible, although some are better than others).

b) Why is this result so much better than the previous result?

5. a) Some programs may cause a "**core dump**". What is a core dump and how can it be caused?

   b) How can it be used to find what's wrong with the program?

6. In the following 2 programs an overflow error occurs. However, in one the answer is completely wrong (negative) and in the other one we see just the message "inf".

```
int main()
{  int x = 2000000000;
   printf("(2 *  %d) / 2  = %d\n", x, (2 *  x) / 2);     // result is -147483648
   printf(" 2 * (%d  / 2) = %d\n", x,  2 * (x  / 2));    // result is 2000000000
}
int main()
{  double x = 1.5e308;
   printf("(2 *  %lf) / 2  = %lf\n", x, (2 *  x) / 2);     // result is inf
   printf(" 2 * (%lf  / 2) = %lf\n", x,  2 * (x  / 2));    // result is 1.5e308
}
```

a) Explain in terms of computer architecture why the results are different.

b) What can we do to report the message "inf" in the first program as well instead of just nonsense?

c) Why don't computers always test for overflow errors, also with int?

7. Consider the following C program on 2 different types of computers:

```
int main()
{
   unsigned long i = 2, j = 3;

   printf("i = %lu (loc. %lu), j = %lu (loc. %lu)\n",
          i, (unsigned long) &i, j, (unsigned long) &j);
}
```

a) What is the effect of the & operator in the code above?

b) The output on computer 1 is as follows:

```
i = 2 (loc. 4291651944), j = 3 (loc. 4291651940)
```

Which conclusions can be drawn about this computer from this result?

c) On another computer 2 we see the following output for the same program:

```
i = 2 (loc. 140737395587568), j = 3 (loc. 140737395587560)
```

What are the differences between this computer 2 and computer 1?

d) If you know that $2^{32} = 4294967296$ and that $2^{47} = 140737488355328$ what more can you say about the architecture on both systems and the way their memory is organized?

8. Consider the following Makefile with 5 labels :

```
# rule 1:
all: myprogram
# rule 2:
myprogram: myprogram.o
# rule 3:
% : %.o
        gcc -o $@ $<
# rule 4:
% : %.c
        gcc -o $@ $<
# rule 5:
%.o : %.c
        gcc -c -o $@ $<
clean:
        rm -f *.o myprogram
```

a) Explain what each rule 1-5 does.

b) Suppose we build an executable from myprogram.c by running "make". Which labels are involved now and in which order?

c) Now we remove the rule with label 2, remove the previous output by running "make clean" and run "make" again to build an executable. What is the difference with the situation in b) ?

9. An executable program `mystery` was compiled on an unknown system and uses some dynamic libraries which are not on your system, so it will fail to run.

   a) How can you find out which libraries are missing ?

   b) What could you do to make the program work without asking for the source code and recompile?

10. The program `optimising.c` below was written by a clumsy programmer and contains at least 5 instructions that are not very efficient. Read it carefully, describe these inefficient instructions and explain why they are not efficient.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define DIMX 4
#define DIMY 5

int main(int argc, char **argv)
{
   int x,y;
   double m,n,o,p,q;
   double (*array)[DIMX]; // Using the modern C99 method for 2D arrays

   array = (double (*)[DIMX]) malloc(DIMX * DIMY * sizeof(double));
   if(array == NULL)
     exit(1);

   p = 0;
   for(x=1;x<100;x++)
      p += x;

   o = pow(p,2);
   printf("p = %lf, o = %lf\n", p, o);

   for(y=0;y<DIMY;y++)
   {  m = 10 * o;
      for(x=0;x<DIMX;x++)
      {  n = 10 * y;
         q = n / 0.25;
         array[y][x]=m + n + o + q + x;
      }
   }
   for(x=0;x<DIMX;x++)
     for(y=0;y<DIMY;y++)
       printf("array[%02d][%02d] = %lf \n",x,y,array[y][x]);

   free(array);
   exit(0);
}
```

Some trigonometric relations:

$\cos(x)^2 + \sin(x)^2 = 1$	$\cos(2x) = \cos(x)^2 - \sin(x)^2$
$\cos(2x) = 1 - 2\sin(x)^2$	$\cos(2x) = 2\cos(x)^2 - 1$
$\sin(2x) = 2\sin(x)\cos(x)$	$\tan(2x) = \dfrac{2\tan(x)}{1-\tan(x)^2}$
$\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$	$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$