

## QUESTIONS TO PREPARE FOR THE FINAL EXAM IN CS 498

### LIBRARY GENERATORS

- 1) Suppose you take a library that has been hand-optimized for a Pentium 4 SSE and execute it in an Athlon XP SSE.
  - a) Is the code portable?
  - b) Is performance portable, i.e., what can be expected of the performance of this code when running on the Pentium 4 versus the performance when running in the Athlon? Justify your answer.
- 2) Explain what is the technique used by library generators to generate libraries that adapt to different target machines.
- 3) How can the technique in 2) be used to generate libraries whose performance depends on input characteristics (like the case of sorting)?
- 4) Many library generators generate C code and then invoke the native compiler to generate the object code.
  - a) Why do library generators follow this strategy?
  - b) Suppose the native compiler does not perform some low level optimizations that are important for the library being generated. What can be done?

### THE ATLAS SYSTEM

- 1) The code generated by the ATLAS system has this form:

```
for (j=1; j<=M; j +=NB)
  for (i=1; i<=N; i +=NB)
    for (k=1; k<=K; k +=NB)
      // mini-MMM code
      for (jj=1; jj<=j+NB; jj+=NU)
        for (ii=1; ii<=i+NB; ii+=MU)
          for (kk=1; kk<=k+NB; kk++)
            // micro-MMM code
            C[ii][jj]+= A[ii][kk] * B[kk][jj]
            C[ii+1][jj]+= A[ii+1][kk] * B[kk][jj]
            C[ii+2][jj]+= A[ii+2][kk] * B[kk][jj]
            C[ii][jj+1]+= A[ii][kk] * B[kk][jj+1]
            C[ii+1][jj+1]+= A[ii+1][kk] * B[kk][jj+1]
            C[ii+2][jj+1]+= A[ii+2][kk] * B[kk][jj+1]
```

In the example above, MU=3 and NU=2

- a) How many floating-point registers are necessary to schedule the micro-MMM code of the figure above, assuming that the architecture has a Fused Multiply-Add instruction?
  - b) How many floating-point registers are necessary to schedule the micro-MMM code if the architecture does not have a Fused Multiply Add and we know that the parameter Latency that ATLAS uses to generate code has the value 2?. Justify your answer.
  - c) How many floating-point registers are necessary to schedule the micro-MMM code if the value of  $KU=5$ , in both cases a) and b)
- 2) Tiles in ATLAS are copied to consecutive memory locations.
- a) Why is this done?
  - b) Explain why copying tiles to consecutive memory locations reduces the number of TLB misses versus non-copying them. Give an example where non-copying the tiles can dramatically increase the number of TLB misses.
- 3) In the case of ATLAS, where tiles are copied to consecutive memory locations, and tiling is applied only to the L1 cache of L1Size:
- a) How can you compute the tile size (NB) so that there are only cold misses?
  - b) How can you compute the tile size (NB) if we assume the cache is fully associative, the line size is 1 word and we use an optimal replacement policy. To answer this question assume that we are using Fortran code, and the loop order is the one in the example below:

```

for (int j = 0; j < N; j++)
  for (int i = 0; i < N; i++)
    for (int k = 0; k < N; k++)
      c[i][j] += a[i][k] * b[k][j]

```

- c) How does the equation above ( in b) changes if we assume that line is  $> 1$ ?
- d) How does the equation above (in c) changes if we assumes that we are using LRU algorithm (not at optimal replacement algorithm)

Notice that the equations are in the class notes, but need to be appropriately justified.

## ITERATIVE COMPILATION

- 1) X language
  - a) What is the goal of this approach?
  - b) Explain two transformations that can be applied with this approach, and show a small example of how they work.

## AUTOMATIC TUNING MATRIX MULTIPLICATION PERFORMANCE ON GRAPHICS HARDWARE

1) List which are the tuning parameters used in the work by Jiang and Snir to generate different versions of matrix-matrix multiplication for the Graphics Hardware.

## SPARSE MATRIX-VECTOR MULTIPLICATION

1) Given the matrix below ( 6X4) where . represent zero values:

$$A = \begin{bmatrix} b & c & . & . \\ a & . & . & . \\ b & c & a & . \\ b & . & . & . \\ . & . & b & c \\ a & . & . & . \end{bmatrix}$$

- a) How would you represent it in Compressed Sparse Row (CSR) Format?
  - b) And in Blocked (BCSR) format assuming  $r=c=2$ ?
- 2) Write the code for a matrix-vector multiplication assuming the CSR format.
- a) Explain what is the problem of this code when compared to a dense matrix-vector multiplication?
- 3) Write the code for matrix-vector multiplication assuming the BCSR format when the block size  $r \times c$  is  $2 \times 2$ .
- a) Is the BCSR format better or worse than CSR? Justify your answer.
- 4) Using the BCSR format,  $r$  and  $c$  can take different values.
- a) what are the architectural features that determine the best values for  $r$  and  $c$ ?
  - b) What are the input characteristics (characteristics of the input matrix) that determine the best values for  $r$  and  $c$ ?

## SORTING LIBRARY

- 1) What makes the problem of generating algorithms for sorting different from generating other types of algorithms (like dense matrix-matrix multiplication). Justify your answer
- 2) What are the two strategies explained in class to build a generator for sorting routines?

## PARALLEL DATA MINING

1) Explain the reason for low scalability when parallelizing the Frequent Pattern Mining algorithm in a cluster of processors. Describe the solution proposed by Cong et al. to solve the problem.

## THE C++ DESIGN TECHNIQUES

1) List two major time overheads associated with the virtual functions in C++.

2) Consider the two codes below. Assume that Base has several derived classes and that vfl is a virtual function. Which one do you think that will run faster? Justify your answer.

```
void func (Base& b)
{
    for (int i = 0; i < 10000; i++)
        for (int j = 0; j < 10000; j++)
            b.vfl();
}

void func (Base& b)
{
    if (type(b) == type(Derived1))
    {
        for (int i = 0; i < 10000; i++)
            for (int j = 0; j < 10000; j++)
                (Derived1) b.Derived1::vfl();
    } else ....{

    }
    ....
    else { // general case
        for (int i = 0; i < 10000; i++)
            for (int j = 0; j < 10000; j++)
                b.vfl();
    }
}
```

## SPIRAL

1) How does SPIRAL generate different versions of a given transform, for example, DFT of size 16?