Introduction to High Performance Computing

April 15, 2019

1 Syllabus

This calendar is subject to change. Anything posted in the future is certainly tentative. Also, anything posted in the recent past may not be correct if I haven’t updated it yet.

1.1 Week 1

1. **Von Neumann Architecture (VNA)**
   - Tue April 23, 2019
     - Motivation for using the ULM (Ulm’s Lecture Machine) as reference model.
     - Representation of unsigned integer numbers as decimal, binary and hexadecimal numerals.
     - Components: RAM, Data bus, CPU.
     - Von Neumann Cycle: How machine code gets executed.
     - Short excerpt of the ULM instruction set (Machine code).

2. **VNA: Conditional Jumps and Integer Arithmetic**
   - Fri April 26, 2019
     - Assembly language for the ULM.
     - Connection to C:
       - pointers and (referenced) values.
       - variable types.
     - Signed integer (two’s complement).
     - Arithmetic logical unit (ALU) and its status flags.
     - Conditional jumps in machine/assembly code

1.2 Week 2

3. **VNA: Control Structures**
   - Tue April 30, 2019
     - Macros for assembly code
     - Example: Stack
     - Control structures (if-then-else, for-loops, while-loops)
4. VNA: Function Calls  
Fr May 3, 2019

- Functions and procedures:
  - Passing parameters (call by value, call by reference).
  - Returning from call.
  - Local and global variables.
- Memory layout of a process: Text segment, data segment, stack and heap.
- Layout of a program file.
- Concept of a linker.

1.3 Week 3

5. Introduction to C (C99)  
Tue May 7, 2019

- How to describe the syntax of a programming language
- Tools for creating a executable program:
  - Preprocessor
  - Tokenizer
  - Parser
  - Assembler
  - Linker
- Differences and similarities between ULM and actual computers:
  - Virtual memory
  - Processes
  - Program loader

6. C99: Control Statements and Functions  
Fr May 10, 2019

- Types and variables
- C control statements: if-then statements and loops.
- Special variable types: Pointers
- Functions
### 1.4 Week 4

<table>
<thead>
<tr>
<th>7.</th>
<th>C99: Arrays, Memory Allocation and Benchmarks</th>
<th>Tue May 14, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Dynamic memory allocation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• BLAS (Basic Linear Algebra Subprogram) Level 1 operations (as application for arrays).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Timer for benchmarks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Gnuplot</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.</th>
<th>C99: Representing matrices</th>
<th>Fr May 17, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Full storage format for matrices (and vectors).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Introducing some BLAS Level 2 Operations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Notes on testing and benchmarking BLAS operations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– General form of a benchmark (that also tests correctness).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Test cases with random numbers and NaNs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Estimating the error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Controlling test parameters through macros</td>
<td></td>
</tr>
</tbody>
</table>

### 1.5 Week 5

<table>
<thead>
<tr>
<th>9.</th>
<th>CPU Caches</th>
<th>Tue May 21, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Concept of CPU caches:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Cache line, cache miss, cache hit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Cache associativity.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Prefetching</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10.</th>
<th>CPU Caches: Optimized matrix vector product</th>
<th>Fr May 24, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Mathematical notation for blocked matrix operations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Simple optimization for row or column major storage.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Advanced optimization using fused vector operations.</td>
<td></td>
</tr>
</tbody>
</table>
### 1.6 Week 6

#### 11. Outlook: Generic and object oriented programming  
**Tue May 28, 2019**

- Namespaces in C99 and C++.
- Generic programming (of functions):
  - In C99 with macros.
  - In C++ with template functions.
- Object oriented programming (OOP)
  - In C99 with `struct` and function pointers.
  - In C++ with `struct` (or `class`) and builtin language features (RAII, methods, operators).
- Combining generic programming and OOP.
- Typical pitfalls (focus on examples from generic programming).

#### 12. General Matrix-Matrix Product (GEMM)  
**Fr May 31, 2019**

- Definition of the GEMM operation.
- Potential for high performance.
- Relevance for other numerical linear algebra operations.
- Some obvious ways to implement the GEMM operation relatively efficient.
- Test and benchmark suite for the GEMM operation.

### 1.7 Week 7

**Tue June 4, 2019**

- Benchmark comparing our GEMM with the Intel MKL implementation.
- Blocked and buffered computation.

#### 14. GEMM: Advanced Cache Optimization  
**Fr June 7, 2019**

- Exploiting the complete cache hierarchy:
  - Frame function and macro/micro kernel for GEMM.
  - Packing matrix blocks.
- Testing components of the overall algorithm.
## 1.8 Week 8

<table>
<thead>
<tr>
<th>15. GEMM: Advanced Cache Optimization</th>
<th>Tue June 11, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Using a profiler: identify code spots where performance gets lost.</td>
<td></td>
</tr>
<tr>
<td>- Instruction pipeline optimizations: Enable compilers to unroll loops.</td>
<td></td>
</tr>
<tr>
<td>- Enable compilers to inline functions.</td>
<td></td>
</tr>
</tbody>
</table>

## 1.9 Week 9

<table>
<thead>
<tr>
<th>17. GEMM: Optimized Micro Kernel</th>
<th>Tue June 18, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Single Instruction Multiple Data (SIMD).</td>
<td></td>
</tr>
<tr>
<td>- GEMM micro kernel in assembly code exploiting SIMD processor features.</td>
<td></td>
</tr>
</tbody>
</table>

## 1.10 Week 10

<table>
<thead>
<tr>
<th>19. ulmBLAS: Providing a Fortran interface</th>
<th>Tue June 25, 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Short introduction of Fortran.</td>
<td></td>
</tr>
<tr>
<td>- Calling Fortran functions from C.</td>
<td></td>
</tr>
<tr>
<td>- Calling C functions from Fortran.</td>
<td></td>
</tr>
</tbody>
</table>
20. **LU Factorization (LU)**  
   - Mathematical background: LU factorization with and without pivoting.
   - Using the factorization to solve linear matrix equations.
   - LU factorization of non-square matrices.
   - Exercise: Setup a test and benchmark suite.

### 1.11 Week 11

21. **LU: Unblocked**  
   - Deriving unblocked variants (i.e. based on BLAS level 2 operations) of the LU factorization.

22. **LU: Blocked**  
   - Deriving blocked variants (i.e. based on BLAS level 3 operations and a unblocked variant) of the LU factorization.

### 1.12 Week 12

23. **Triangular Solvers (TRS)**  

24. **TRS: Unblocked**

### 1.13 Week 13

25. **TRS: Blocked**

26. **TRS: Blocked**

### 1.14 Week 14

27. **Final Benchmarks and Outlook on Multithreading**
| 28. Final Benchmarks and Outlook on Multithreading | Fr July 26, 2019 |