

---

# Contents

---

## Preface

xiii

## 1 C/C++: Review

1

1.1	An example: The Aitken transformation	3
1.1.1	Leibniz series and the logarithmic series	4
1.1.2	Modular organization of sources	6
1.2	C review	9
1.2.1	Header files	10
1.2.2	Arrays and pointers	12
1.2.3	The Aitken iteration using arrays and pointers	17
1.2.4	Declarations and definitions	20
1.2.5	Function calls and the compilation process	21
1.3	C++ review	27
1.3.1	The <code>Vector</code> class	28
1.3.2	Aitken transformation in C++	48
1.4	A little Fortran	52
1.5	References	55

## 2 C/C++: Libraries and Makefiles

57

2.1	Mixed-language programming	59
2.1.1	Transmutation of names from source to object files	60

2.1.2	Linking Fortran programs with C and C++	64
2.2	Using BLAS and LAPACK libraries	69
2.2.1	Arrays, matrices, and leading dimensions	71
2.2.2	BLAS and LAPACK	74
2.2.3	C++ class interface to BLAS/LAPACK	77
2.3	Building programs using GNU Make	81
2.3.1	The <code>utils/</code> folder	84
2.3.2	Targets, prerequisites, and dependency graphs	87
2.3.3	Make variables in <code>makevars.mk</code>	92
2.3.4	Pattern rules in <code>makevars.mk</code>	94
2.3.5	Phony targets in <code>makevars.mk</code>	97
2.3.6	Recursive <code>make</code> and <code>.d</code> files	98
2.3.7	Beyond recursive <code>make</code>	103
2.3.8	Building your own library	106
2.4	The Fast Fourier Transform	113
2.4.1	The FFT algorithm in outline	115
2.4.2	FFT using MKL	117
2.4.3	FFT using FFTW	120
2.4.4	Cycles and histograms	122
2.4.5	Optimality of FFT implementations	125
2.5	References	130
<b>3</b>	<b>The Processor</b>	<b>133</b>
3.1	Overview of the x86 architecture	139
3.1.1	64-bit x86 architecture	139
3.1.2	64-bit x86 assembly programming	144
3.1.3	The Time Stamp Counter	150
3.1.4	Cache parameters and the <code>CPUID</code> instruction	152
3.2	Compiler optimizations	156
3.2.1	Preliminaries	157
3.2.2	Loop unrolling	160
3.2.3	Loop fusion	168
3.2.4	Unroll and jam	170
3.2.5	Loop interchange	173
3.2.6	C++ overhead	178



3.2.7	A little compiler theory . . . . .	180
3.3	Optimizing for the instruction pipeline . . . . .	186
3.3.1	Instruction pipelines . . . . .	188
3.3.2	Chipsets . . . . .	193
3.3.3	Peak floating point performance . . . . .	196
3.3.4	Microkernel for matrix multiplication . . . . .	209
3.4	References . . . . .	219
<b>4</b>	<b>Memory</b>	<b>221</b>
4.1	DRAM and cache memory . . . . .	224
4.1.1	DRAM memory . . . . .	226
4.1.2	Cache memory . . . . .	230
4.1.3	Physical memory and virtual memory . . . . .	234
4.1.4	Latency to DRAM memory: First attempts . . . . .	237
4.1.5	Latency to DRAM . . . . .	241
4.2	Optimizing memory access . . . . .	246
4.2.1	Bandwidth to DRAM . . . . .	248
4.2.2	Matrix transpose . . . . .	251
4.2.3	Optimized matrix multiplication . . . . .	256
4.3	Reading from and writing to disk . . . . .	266
4.3.1	C versus C++ . . . . .	268
4.3.2	Latency to disk . . . . .	270
4.3.3	Bandwidth to disk . . . . .	273
4.4	Page tables and virtual memory . . . . .	274
4.4.1	Partitioning the virtual address space . . . . .	276
4.4.2	Physical address space and page tables . . . . .	279
4.5	References . . . . .	284
<b>5</b>	<b>Threads and Shared Memory</b>	<b>287</b>
5.1	Introduction to OpenMP . . . . .	290
5.1.1	OpenMP syntax . . . . .	290
5.1.2	Shared variables and OpenMP's memory model . . . . .	297
5.1.3	Overheads of OpenMP constructs . . . . .	299
5.2	Optimizing OpenMP programs . . . . .	304
5.2.1	Near memory and far memory . . . . .	305



5.2.2	Bandwidth to DRAM memory . . . . .	308
5.2.3	Matrix transpose . . . . .	310
5.2.4	Fast Fourier transform . . . . .	312
5.3	Introduction to Pthreads . . . . .	316
5.3.1	Pthreads . . . . .	317
5.3.2	Overhead of thread creation . . . . .	324
5.3.3	Parallel regions using Pthreads . . . . .	326
5.4	Program memory . . . . .	348
5.4.1	An easy system call . . . . .	349
5.4.2	Stacks . . . . .	353
5.4.3	Segmentation faults and memory errors . . . . .	357
5.5	References . . . . .	367
<b>6</b>	<b>Special Topic: Networks and Message Passing</b>	<b>369</b>
6.1	MPI: Getting started . . . . .	371
6.1.1	Initializing MPI . . . . .	373
6.1.2	Unsafe communication in MPI . . . . .	376
6.2	High-performance network architecture . . . . .	380
6.2.1	Fat-tree network . . . . .	382
6.2.2	Infiniband network architecture . . . . .	384
6.3	MPI examples . . . . .	399
6.3.1	Variants of MPI send and receive . . . . .	401
6.3.2	Jacobi iteration . . . . .	416
6.3.3	Matrix transpose . . . . .	427
6.3.4	Collective communication . . . . .	452
6.3.5	Parallel I/O in MPI . . . . .	461
6.4	The Internet . . . . .	472
6.4.1	IP addresses . . . . .	475
6.4.2	Send and receive . . . . .	481
6.4.3	Server . . . . .	482
6.4.4	Client . . . . .	486
6.4.5	Internet latency . . . . .	489
6.4.6	Internet bandwidth . . . . .	491
6.5	References . . . . .	495

<b>7</b>	<b>Special Topic: The Xeon Phi Coprocessor</b>	<b>497</b>
7.1	Xeon Phi architecture . . . . .	501
7.1.1	Peak floating point bandwidth . . . . .	502
7.1.2	A simple Phi program . . . . .	502
7.1.3	Xeon Phi memory system . . . . .	505
7.2	Offload . . . . .	508
7.2.1	Initializing to use the MIC device . . . . .	509
7.2.2	The <code>target(mic)</code> declaration specification . . . . .	511
7.2.3	Summing the Leibniz series . . . . .	513
7.2.4	Offload bandwidth . . . . .	522
7.3	Two examples: FFT and matrix multiplication . . . . .	524
7.3.1	FFT . . . . .	524
7.3.2	Matrix multiplication . . . . .	526
<b>8</b>	<b>Special Topic: Graphics Coprocessor Programming Using CUDA</b>	<b>535</b>
8.1	Graphics coprocessor architecture . . . . .	538
8.1.1	Graphics processor capability . . . . .	538
8.1.2	Host and device memory . . . . .	541
8.1.3	Timing CUDA kernels . . . . .	544
8.1.4	Warps and thread blocks . . . . .	546
8.2	Introduction to CUDA . . . . .	551
8.2.1	Summing the Leibniz series . . . . .	552
8.2.2	CUDA compilation . . . . .	563
8.3	Two examples . . . . .	566
8.3.1	Bandwidth to memory . . . . .	567
8.3.2	Matrix multiplication . . . . .	568
8.4	References . . . . .	576
<b>A</b>	<b>Machines Used, Plotting, Python, GIT, Cscope, and gcc</b>	<b>577</b>
A.1	Machines used . . . . .	578
A.2	Plotting in C/C++ and other preliminaries . . . . .	578
A.3	C/C++ versus Python versus MATLAB . . . . .	582
A.4	GIT . . . . .	584
A.5	Cscope . . . . .	585
A.6	Compiling with gcc/g++ . . . . .	586