

# 3.

## The microarchitecture of Intel, AMD and VIA CPUs

### An optimization guide for assembly programmers and compiler makers

By Agner Fog. Technical University of Denmark.  
Copyright © 1996 - 2016. Last updated 2016-01-16.

#### Contents

1	Introduction .....	5
1.1	About this manual .....	5
1.2	Microprocessor versions covered by this manual .....	6
2	Out-of-order execution (All processors except P1, PMMX) .....	9
2.1	Instructions are split into $\mu$ ops .....	9
2.2	Register renaming .....	10
3	Branch prediction (all processors) .....	12
3.1	Prediction methods for conditional jumps .....	12
3.2	Branch prediction in P1 .....	18
3.3	Branch prediction in PMMX, PPro, P2, and P3 .....	21
3.4	Branch prediction in P4 and P4E .....	23
3.5	Branch prediction in PM and Core2 .....	25
3.6	Branch prediction in Intel Nehalem .....	27
3.7	Branch prediction in Intel Sandy Bridge and Ivy Bridge .....	28
3.8	Branch prediction in Intel Haswell, Broadwell and Skylake .....	29
3.9	Branch prediction in Intel Atom and Silvermont .....	29
3.10	Branch prediction in VIA Nano .....	30
3.11	Branch prediction in AMD K8 and K10 .....	31
3.12	Branch prediction in AMD Bulldozer, Piledriver and Steamroller .....	33
3.13	Branch prediction in AMD Bobcat and Jaguar .....	34
3.14	Indirect jumps on older processors .....	35
3.15	Returns (all processors except P1) .....	35
3.16	Static prediction .....	36
3.17	Close jumps .....	37
4	Pentium 1 and Pentium MMX pipeline .....	38
4.1	Pairing integer instructions .....	38
4.2	Address generation interlock .....	42
4.3	Splitting complex instructions into simpler ones .....	42
4.4	Prefixes .....	43
4.5	Scheduling floating point code .....	44
5	Pentium 4 (NetBurst) pipeline .....	47
5.1	Data cache .....	47
5.2	Trace cache .....	47
5.3	Instruction decoding .....	52
5.4	Execution units .....	53
5.5	Do the floating point and MMX units run at half speed? .....	56
5.6	Transfer of data between execution units .....	58
5.7	Retirement .....	61
5.8	Partial registers and partial flags .....	61
5.9	Store forwarding stalls .....	62
5.10	Memory intermediates in dependency chains .....	62
5.11	Breaking dependency chains .....	64
5.12	Choosing the optimal instructions .....	64
5.13	Bottlenecks in P4 and P4E .....	67

6	Pentium Pro, II and III pipeline.....	70
6.1	The pipeline in PPro, P2 and P3 .....	70
6.2	Instruction fetch .....	70
6.3	Instruction decoding.....	71
6.4	Register renaming .....	75
6.5	ROB read.....	75
6.6	Out of order execution .....	79
6.7	Retirement .....	80
6.8	Partial register stalls.....	81
6.9	Store forwarding stalls .....	84
6.10	Bottlenecks in PPro, P2, P3 .....	85
7	Pentium M pipeline.....	87
7.1	The pipeline in PM .....	87
7.2	The pipeline in Core Solo and Duo .....	88
7.3	Instruction fetch .....	88
7.4	Instruction decoding.....	88
7.5	Loop buffer .....	90
7.6	Micro-op fusion .....	90
7.7	Stack engine.....	92
7.8	Register renaming .....	94
7.9	Register read stalls .....	94
7.10	Execution units .....	96
7.11	Execution units that are connected to both port 0 and 1.....	96
7.12	Retirement .....	98
7.13	Partial register access.....	98
7.14	Store forwarding stalls .....	100
7.15	Bottlenecks in PM .....	100
8	Core 2 and Nehalem pipeline .....	103
8.1	Pipeline.....	103
8.2	Instruction fetch and predecoding .....	103
8.3	Instruction decoding.....	106
8.4	Micro-op fusion .....	106
8.5	Macro-op fusion .....	107
8.6	Stack engine.....	108
8.7	Register renaming .....	109
8.8	Register read stalls .....	109
8.9	Execution units .....	110
8.10	Retirement .....	114
8.11	Partial register access.....	114
8.12	Store forwarding stalls .....	116
8.13	Cache and memory access.....	117
8.14	Breaking dependency chains .....	118
8.15	Multithreading in Nehalem .....	118
8.16	Bottlenecks in Core2 and Nehalem.....	119
9	Sandy Bridge and Ivy Bridge pipeline .....	121
9.1	Pipeline.....	121
9.2	Instruction fetch and decoding .....	121
9.3	μop cache .....	122
9.4	Loopback buffer .....	124
9.5	Micro-op fusion .....	124
9.6	Macro-op fusion .....	124
9.7	Stack engine.....	125
9.8	Register allocation and renaming.....	126
9.9	Register read stalls .....	127
9.10	Execution units .....	127
9.11	Partial register access.....	131
9.12	Transitions between VEX and non-VEX modes .....	131
9.13	Cache and memory access.....	132

9.14	Store forwarding stalls .....	133
9.15	Multithreading .....	133
9.16	Bottlenecks in Sandy Bridge and Ivy Bridge.....	134
10	Haswell and Broadwell pipeline .....	136
10.1	Pipeline.....	136
10.2	Instruction fetch and decoding .....	136
10.3	μop cache .....	136
10.4	Loopback buffer .....	137
10.5	Micro-op fusion .....	137
10.6	Macro-op fusion .....	137
10.7	Stack engine .....	138
10.8	Register allocation and renaming.....	138
10.9	Execution units .....	139
10.10	Partial register access.....	142
10.11	Cache and memory access.....	143
10.12	Store forwarding stalls .....	144
10.13	Multithreading .....	145
10.14	Bottlenecks in Haswell and Broadwell.....	145
11	Skylake pipeline .....	148
11.1	Pipeline.....	148
11.2	Instruction fetch and decoding .....	148
11.3	μop cache .....	148
11.4	Loopback buffer .....	149
11.5	Micro-op fusion .....	149
11.6	Macro-op fusion .....	149
11.7	Stack engine .....	150
11.8	Register allocation and renaming.....	150
11.9	Execution units .....	151
11.10	Partial register access.....	154
11.11	Cache and memory access.....	155
11.12	Store forwarding stalls .....	156
11.13	Multithreading .....	156
11.14	Bottlenecks in Skylake .....	156
12	Intel Atom pipeline.....	159
12.1	Instruction fetch .....	159
12.2	Instruction decoding.....	159
12.3	Execution units .....	159
12.4	Instruction pairing.....	160
12.5	X87 floating point instructions .....	161
12.6	Instruction latencies .....	161
12.7	Memory access.....	162
12.8	Branches and loops .....	163
12.9	Multithreading .....	163
12.10	Bottlenecks in Atom .....	164
13	Intel Silvermont pipeline .....	164
13.1	Pipeline.....	165
13.2	Instruction fetch and decoding .....	165
13.3	Loop buffer .....	166
13.4	Macro-op fusion .....	166
13.5	Register allocation and out of order execution .....	166
13.6	Special cases of independence.....	166
13.7	Execution units .....	166
13.8	Partial register access.....	167
13.9	Cache and memory access.....	167
13.10	Store forwarding.....	168
13.11	Multithreading .....	168
13.12	Bottlenecks in Silvermont.....	168
14	VIA Nano pipeline.....	170

14.1 Performance monitor counters .....	170
14.2 Instruction fetch .....	170
14.3 Instruction decoding .....	170
14.4 Instruction fusion.....	170
14.5 Out of order system .....	171
14.6 Execution ports .....	171
14.7 Latencies between execution units .....	172
14.8 Partial registers and partial flags .....	174
14.9 Breaking dependence .....	174
14.10 Memory access.....	175
14.11 Branches and loops .....	175
14.12 VIA specific instructions .....	175
14.13 Bottlenecks in Nano .....	176
15 AMD K8 and K10 pipeline .....	177
15.1 The pipeline in AMD K8 and K10 processors .....	177
15.2 Instruction fetch .....	179
15.3 Predecoding and instruction length decoding.....	179
15.4 Single, double and vector path instructions .....	180
15.5 Stack engine .....	181
15.6 Integer execution pipes .....	181
15.7 Floating point execution pipes.....	181
15.8 Mixing instructions with different latency .....	183
15.9 64 bit versus 128 bit instructions .....	184
15.10 Data delay between differently typed instructions.....	185
15.11 Partial register access.....	185
15.12 Partial flag access.....	186
15.13 Store forwarding stalls .....	186
15.14 Loops.....	187
15.15 Cache .....	187
15.16 Bottlenecks in AMD K8 and K10 .....	189
16 AMD Bulldozer, Piledriver and Steamroller pipeline.....	190
16.1 The pipeline in AMD Bulldozer, Piledriver and Steamroller .....	190
16.2 Instruction fetch .....	191
16.3 Instruction decoding .....	191
16.4 Loop buffer .....	192
16.5 Instruction fusion.....	192
16.6 Stack engine .....	192
16.7 Out-of-order schedulers .....	192
16.8 Integer execution pipes .....	193
16.9 Floating point execution pipes.....	193
16.10 AVX instructions.....	194
16.11 Data delay between different execution domains .....	195
16.12 Instructions that use no execution units .....	196
16.13 Partial register access.....	197
16.14 Partial flag access.....	197
16.15 Dependency-breaking instructions .....	197
16.16 Branches and loops .....	198
16.17 Cache and memory access.....	198
16.18 Store forwarding stalls .....	199
16.19 Bottlenecks in AMD Bulldozer, Piledriver and Steamroller .....	200
16.20 Literature .....	202
17 AMD Bobcat and Jaguar pipeline .....	202
17.1 The pipeline in AMD Bobcat and Jaguar .....	202
17.2 Instruction fetch .....	203
17.3 Instruction decoding .....	203
17.4 Single, double and complex instructions .....	203
17.5 Integer execution pipes .....	203
17.6 Floating point execution pipes.....	203

17.7	Mixing instructions with different latency .....	204
17.8	Dependency-breaking instructions .....	204
17.9	Data delay between differently typed instructions .....	204
17.10	Partial register access.....	204
17.11	Cache .....	204
17.12	Store forwarding stalls .....	205
17.13	Bottlenecks in Bobcat and Jaguar .....	205
17.14	Literature: .....	206
18	Comparison of microarchitectures .....	206
18.1	The AMD K8 and K10 kernel .....	206
18.2	The AMD Bulldozer, Piledriver and Steamroller kernel.....	207
18.3	The Pentium 4 kernel.....	208
18.4	The Pentium M kernel.....	210
18.5	Intel Core 2 and Nehalem microarchitecture .....	210
18.6	Intel Sandy Bridge and later microarchitectures .....	211
19	Comparison of low power microarchitectures .....	212
19.1	Intel Atom microarchitecture .....	212
19.2	VIA Nano microarchitecture .....	212
19.3	AMD Bobcat microarchitecture.....	213
19.4	Conclusion.....	213
20	Future trends.....	215
21	Literature.....	218
22	Copyright notice .....	218

# 1 Introduction

## 1.1 About this manual

This is the third in a series of five manuals:

1. Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms.
2. Optimizing subroutines in assembly language: An optimization guide for x86 platforms.
3. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers.
4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs.
5. Calling conventions for different C++ compilers and operating systems.

The latest versions of these manuals are always available from [www.agner.org/optimize](http://www.agner.org/optimize). Copyright conditions are listed on page 218 below.

The present manual describes the details of the microarchitectures of x86 microprocessors from Intel and AMD. The Itanium processor is not covered. The purpose of this manual is to enable assembly programmers and compiler makers to optimize software for a specific microprocessor. The main focus is on details that are relevant to calculations of how much time a piece of code takes to execute, such as the latencies of different execution units and the throughputs of various parts of the pipelines. Branch prediction algorithms are also covered in detail.

[Click here to download full PDF material](#)