

Table of Contents

1	Introduction	1
2	Kinds of Parallelism	3
2.1	Classification by Structural Items	4
2.2	Classification by Items Processed	5
2.3	Classification by Instruction and Data Stream Parallelism	6
2.4	Classification by Memory Organization	7
3	Architectures for Fine-Grain Parallelism	8
3.1	Vertical Parallelism	8
3.2	Horizontal Parallelism	9
3.2.1	VLIW Architecture	9
3.2.2	Superscalar Processor Architectures	11
3.3	Classification of Pipelined Processor Architectures	12
3.4	Other Architectures for Fine-Grain Parallelism	15
3.4.1	Conditional Execution	16
3.4.2	History	17
4	VLIW Machines	18
4.1	Multiflow TRACE	18
4.2	IBM's VLIW Processor	19
4.3	Cydra	22
4.4	LIFE (Philips/Signetics)	24
4.5	The XIMD Architecture	26
5	Constraints on VLIW Architectures	28
6	Architectural Support for Exploitation of Fine-Grain Parallelism	32
6.1	Dynamic Instruction-Scheduling	32
7	Constraints for Instruction Scheduling	38
7.1	Data Dependences	38
7.2	Off-live Dependence	38
8	Instruction-Scheduling Methods	40
8.1	Local Instruction-Scheduling	42

8.2	Global Instruction-Scheduling	42
8.3	Global Scheduling Methods Based on Local Scheduling	43
8.3.1	Trace Scheduling	43
8.3.2	Other Global Scheduling Methods	46
8.4	Global Scheduling on the Program Graph	47
8.4.1	Percolation Scheduling	49
8.4.1.1	Move_op.....	50
8.4.1.2	Move_cj.....	51
8.4.1.3	Unify.....	52
8.4.1.4	Delete.....	53
8.4.2	Extensions to Percolation Scheduling	54
8.5	Loop Parallelization	55
8.5.1	Loop Unrolling	56
8.5.2	Software Pipelining	58
8.5.3	Perfect Pipelining	59
8.5.4	Integrating Loop Parallelization and Instruction Scheduling	61
9	Developing Instruction-Scheduling Methods	62
10	Tools for Instruction Scheduling	63
10.1	The C Compiler	64
11	The Machine Model	65
12	The Horizontal Instruction-Scheduler	67
12.1	The Interface Compiler - Scheduler	67
12.2	The Scheduler's Front-end	68
12.2.1	Mapping rtls to Machine Instructions	70
12.2.2	Flow Graph, Pointer Targets and Call Graph	74
12.2.3	Registers and Register Renaming	75
12.2.4	Memory-Access Analysis	81
12.2.5	Data-Dependence Analysis	90
12.2.6	Inserting Representatives for Multicycle Operations	92
12.2.7	Data-Flow Analysis	96
12.2.8	Standard Optimizations	101
12.3	The Scheduler's Central Part	103
12.3.1	Loop Parallelization	104
12.3.2	Partitioning the Program Graph into Regions	104
12.3.3	Extended Percolation Scheduling - Core Transformations	106

- 12.3.3.1 Standard Core Transformations..... 106
 - 12.3.3.2 Core Transformations for Multicycle Operations 110
 - 12.3.4 Extended Percolation Scheduling - Structure 115
 - 12.3.5 Extended Percolation Scheduling - Control Tactics 117
 - 12.3.5.1 Node-Oriented Tactics 117
 - 12.3.5.2 Operation-Oriented Tactics 120
 - 12.3.5.3 Which Tactics for what Application? 121
 - 12.3.5.4 Control Tactics for Specific Code Patterns 121
 - 12.3.5.5 Control Tactics Considering Resources 123
 - 12.3.6 Extended Percolation Scheduling - Control Strategies 124
 - 12.3.7 Moving Operations out of Regions and Functions 125
 - 12.3.8 Resource Management 128
- 12.4 The Scheduler’s Back-End 129
 - 12.4.1 Register Allocation 129
 - 12.4.2 Interactions Between Register Allocation and Instruction Scheduling 134
 - 12.4.3 Rescheduling 135
 - 12.4.4 Assignment of Operations to PEs 137
 - 12.4.5 Instruction-Word Generation 138
 - 12.4.6 Debug Information 140

13 Resource Management 142

- 13.1 Machine Description 144
- 13.2 Operators and Functions for Resource Management 146
- 13.3 Resource Allocation 147
- 13.4 Resource Assignment 148
 - 13.4.1 Assignment of Processing Elements 148
 - 13.4.2 Assignment of General Resources 152

14 Exceptions 156

15 Vertical Instruction-Scheduling 158

- 15.1 Vertical Core Transformations 159

16 Conclusion 164

17 References 165