Trimaran ILP Reading List By Topical Category

This is not intended to be a comprehensive and exhaustive bibliography of papers in the area of instruction-level parallel processing (ILP). Rather, this list concentrates on those papers that best demonstrate the EPIC architectural philosophy advocated by HPL-PD and the compiler philosophy, framework and techniques embodied in Trimaran. This reading list is designed for people who wish to get up to speed on the use of HPL-PD and Trimaran, and who wish to further the state of the art of compiling for EPIC architectures.

A good starting point is to understand HPL-PD [1], the way in which the architectural features of HPL-PD were intended to be used [2], and the space of EPIC processors that can be described to, and compiled by, Trimaran [3, 4]. Thereafter, one could focus on those topics listed below that represent one’s areas of research interest.

**ILP survey**
A survey of instruction-level parallel processing as of 1992 [5].

**VLIW architecture**
Commercial VLIW products [6-9].

**EPIC architecture**
The original specification of the EPIC style of architecture [1] and a discussion of the underlying philosphy as well as the motivation for the various architectural features [2]. Partial details of the IA-64, the first commercial instance of an EPIC architecture [10].

**ILP compiler systems**
ILP compilers of significant scope [11-14].

**ILP optimizations**
Various optimizations that have special relevance to an ILP processor [11, 15, 16, 12-14, 17, 18].

**Acyclic scheduling**
Trace scheduling [19, 13, 20], superblock scheduling [12], hyperblock scheduling [21], more general scheduling algorithms [22-24], and inter-region scheduling [11, 13, 25].

**Modulo scheduling**
Array variable promotion and expanded virtual registers (EVRs)
Array variable promotion to eliminate array loads and stores that are redundant within or across iterations of the loop [38-40, 14]. The following papers also discuss the value flow analysis required [39, 14, 41] and the use of expanded virtual registers (EVRs) to avoid the premature introduction of register-register copy operations [39, 14].

Register allocation
Rotating register allocation [42]. Register allocation for predicated code [43, 44].

If-conversion and the use of predicates
If-conversion to form predicated code [26, 45, 14]. The application of predicates: an overview [2], in modulo scheduling [26, 27], in scheduling acyclic regions [21, 46, 47], in reducing the length of the critical path through the computation [48, 49], and as an intermediate form when scheduling control-intensive regions for a processor without predicated execution [50]. The benefits of predicated execution in modulo scheduling [51] and in acyclic regions [52].

Analysis of predicated code
Analysis of predicated code [53]. The application of such analyses to register allocation [43, 44].

Control speculation and recovery
Control speculation, i.e., the movement of operations to an earlier point than the branch or predicate-setting operation upon which it is dependent [19, 54, 46, 2, 47] and architectural and compiler techniques to support correct exception handling [55, 54, 56, 2].

Data speculation
Data speculation, i.e., the movement of a load to an earlier point than a store upon which it might be dependent [57-59, 2].

Tolerating branch latency
"Unbundled" branches [2]. Static branch prediction [60, 61]. Programmatically-computed dynamic branch prediction [62].

Critical path reduction (CPR)
Compiler transformations to reduce the length of the critical path through a computation [63, 48, 49, 64, 65].

Tolerating data cache miss latency
Predicting which memory references will cause data cache misses and either prefetching them or scheduling them early [66] [67] [68] [69] [2].
**Machine description**

The machine description database and query interface used by the ILP compiler to understand the relevant details of the target processor [3, 4, 70, 71]. Optimization of the machine description [72, 73].

**Region-based compiling**

An approach to structuring ILP compilers which is embedded in, and supported by, Trimaran [18].

**Note:** A number of the HPL Technical Reports in this reading list have been published as journal articles or in conference proceedings. The technical reports were listed because they are more complete and comprehensive than the corresponding papers. However, for the benefit of individuals who would like to read the paper, and for the purposes of referencing these works, we list here pairs of references. In each pair, the first reference is the technical report and the second one is the corresponding paper:

[5], [74]
[27], [75]
[28], [74]
[49], [76]
[53], [77]
[48], [78]
[64], [65]
[70], [71]
Bibliography


52. S. A. Mahlke, R. E. Hank, R. A. Bringmann, J. C. Gyllenhaal, D. M. Gallagher and W. W. Hwu. Characterizing the impact of predicated execution on branch


