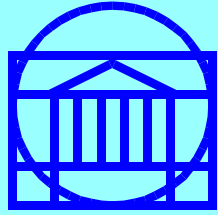


The Zephyr Compiler Infrastructure



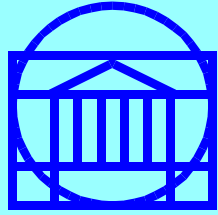
**Princeton University
University of Virginia**



The Problem

University of Virginia

- ◆ **Inadequate compiler infrastructure impedes progress in high-performance computing**
 - ◆ **Compiler development currently lags hardware development cycle**
 - ◆ **Inadequate infrastructure traps systems software on outdated, expensive hardware**
 - ◆ **Barrier to entry in compiler research is high**
 - ◆ **Difficult to share, reuse, or combine software resulting from research efforts**



The Zephyr Project

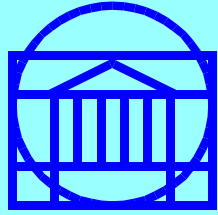
University of Virginia

◆ **Goal**

- ◆ **Deliver high-quality tools for rapidly constructing compilers for experimental computing systems research**

◆ **How**

- ◆ **Provide specification languages and processors to automatically generate key compiler components**
 - **Don't write code, write specifications!**

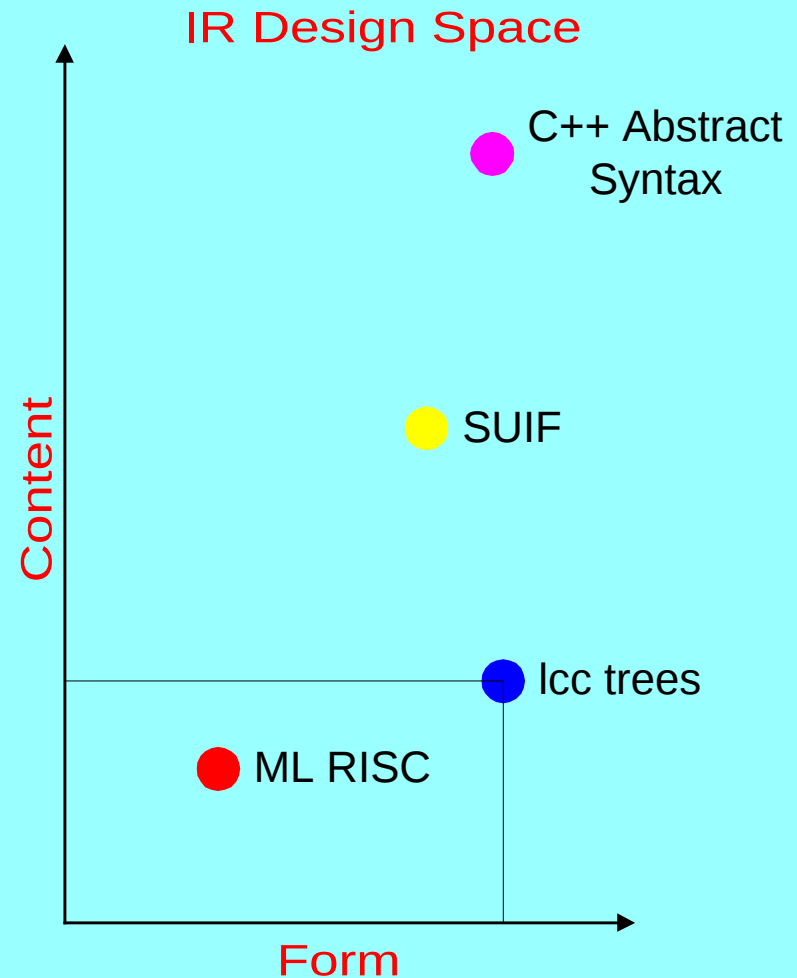


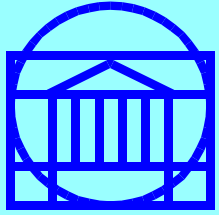
The Zephyr Approach

University of Virginia

◆ Intermediate representations (IRs)

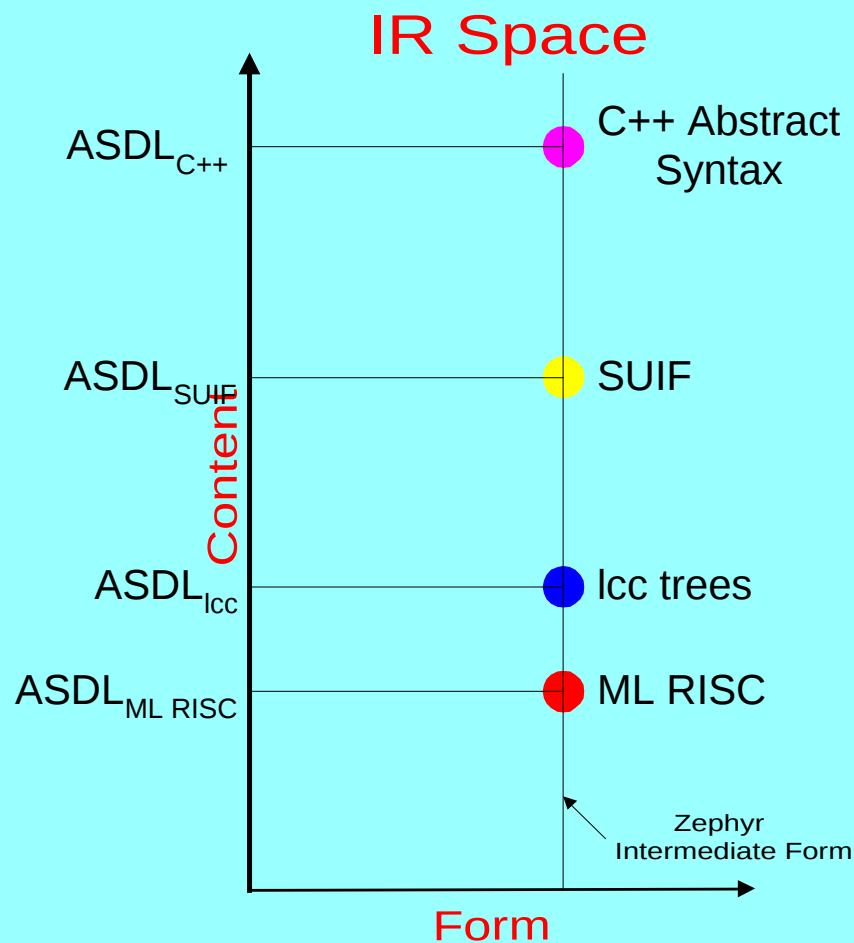
- ◆ Form and content are inseparable
- ◆ Fixed point in the design space
- ◆ Cannot reuse tools
- ◆ IR may not suit needs





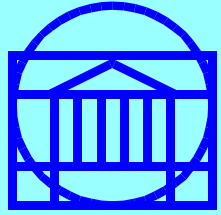
The Zephyr Approach

University of Virginia



◆ Separate form from content

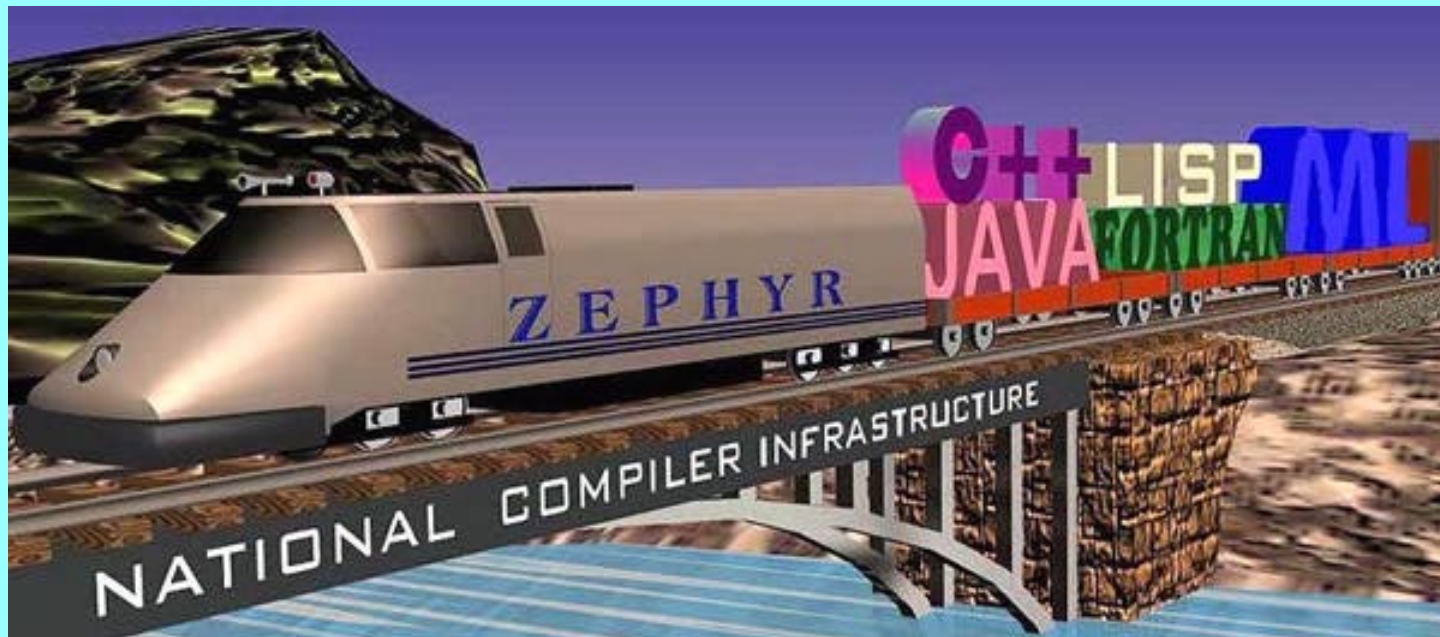
- ◆ Fixed intermediate form: ZIF (trees)
- ◆ Content specified using ASDL and CSDL

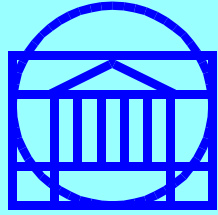


University of Virginia

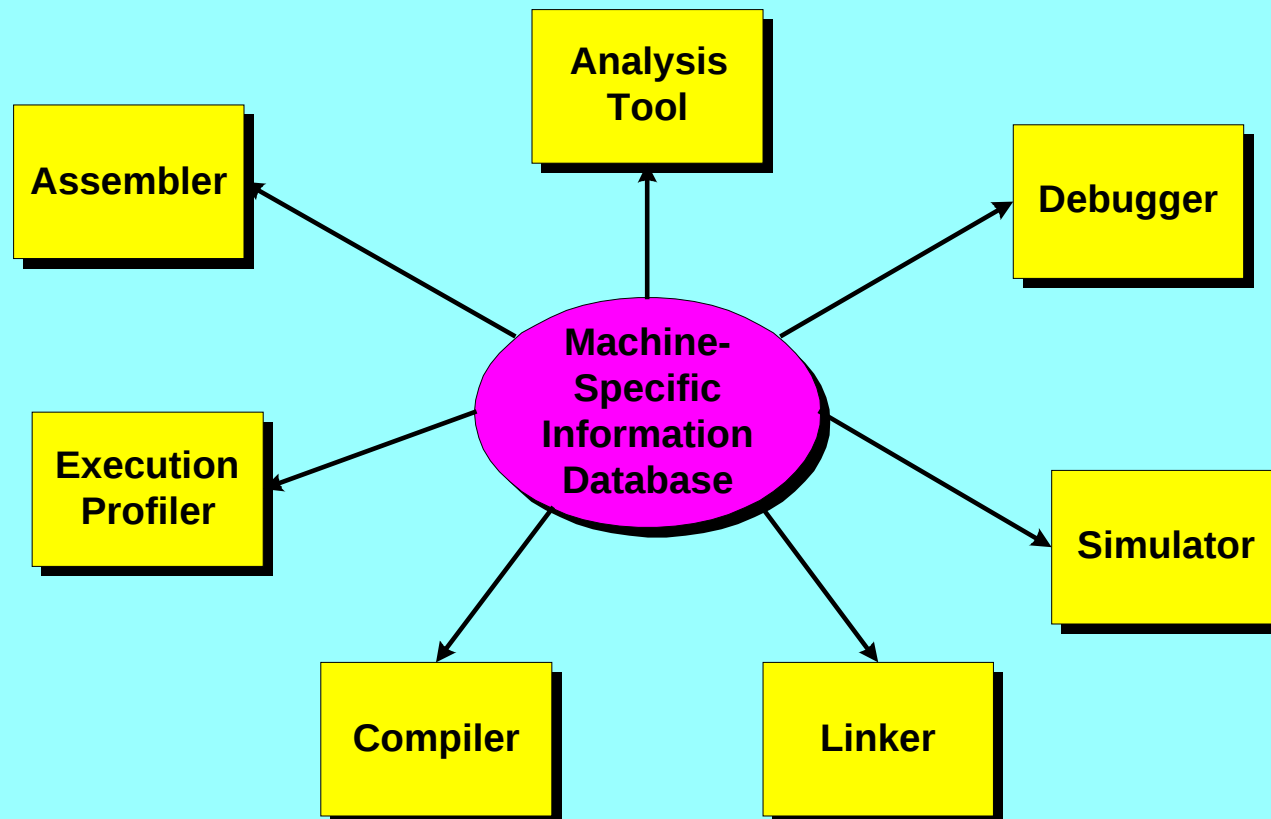
Key Zephyr Building Blocks

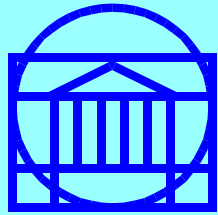
- ◆ **ASDL - Abstract Syntax Description Language**
- ◆ **CSDL - Computer System Description Language**
- ◆ **VPO - Very Portable Optimizer**





- ◆ **Vision - To have a single application-independent description**

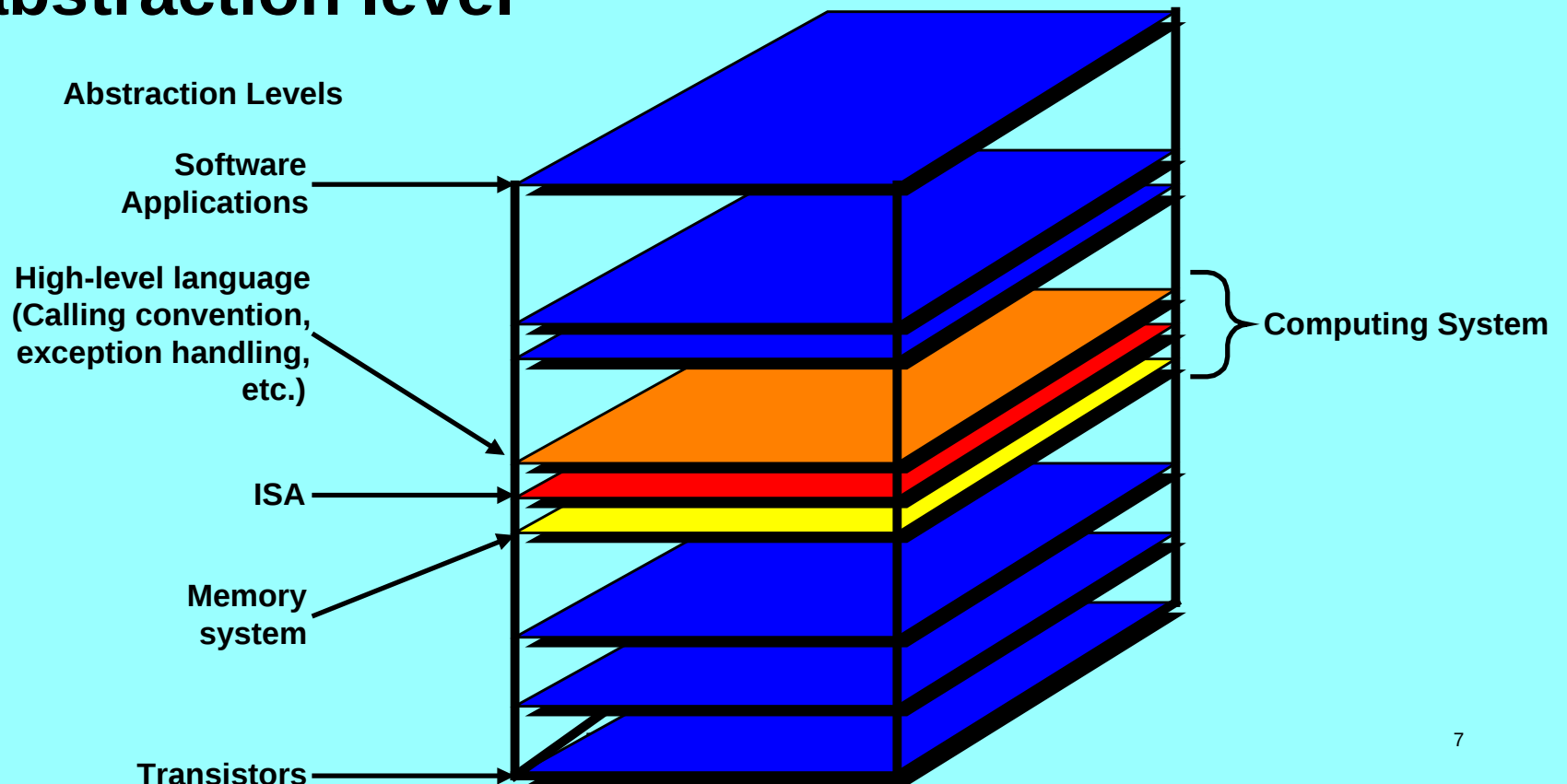


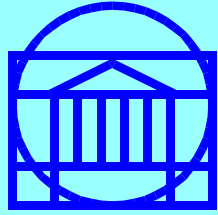


Machine Abstraction Levels

University of Virginia

- ◆ The *computing system* level of abstraction reaches above and below the ISA abstraction level



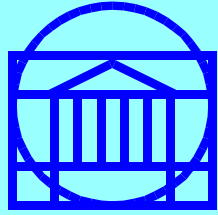


Solution

University of Virginia

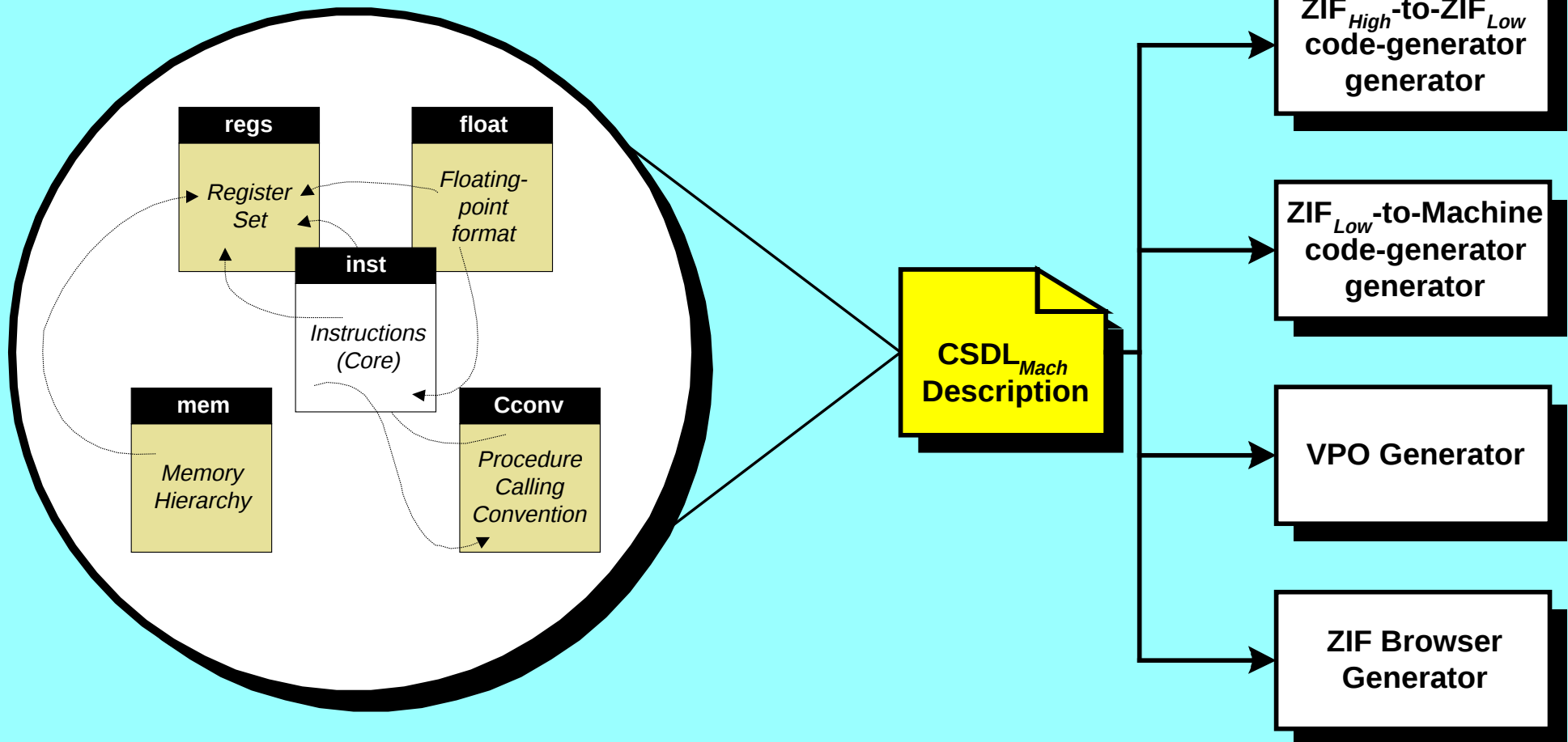
◆ **Computing System Description Language**

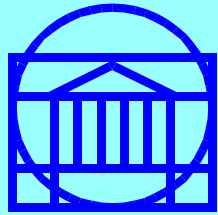
- ◆ **Modular system of components**
 - Allow user to customize a description
 - A specification can be incomplete
- ◆ **Easily extensible for adding new details**
- ◆ **Facilitates reusability/application-independence of descriptions**
- ◆ **Fully typed representation**
 - type checkers
 - prohibit nonsensical specifications



Generation of target-specific components

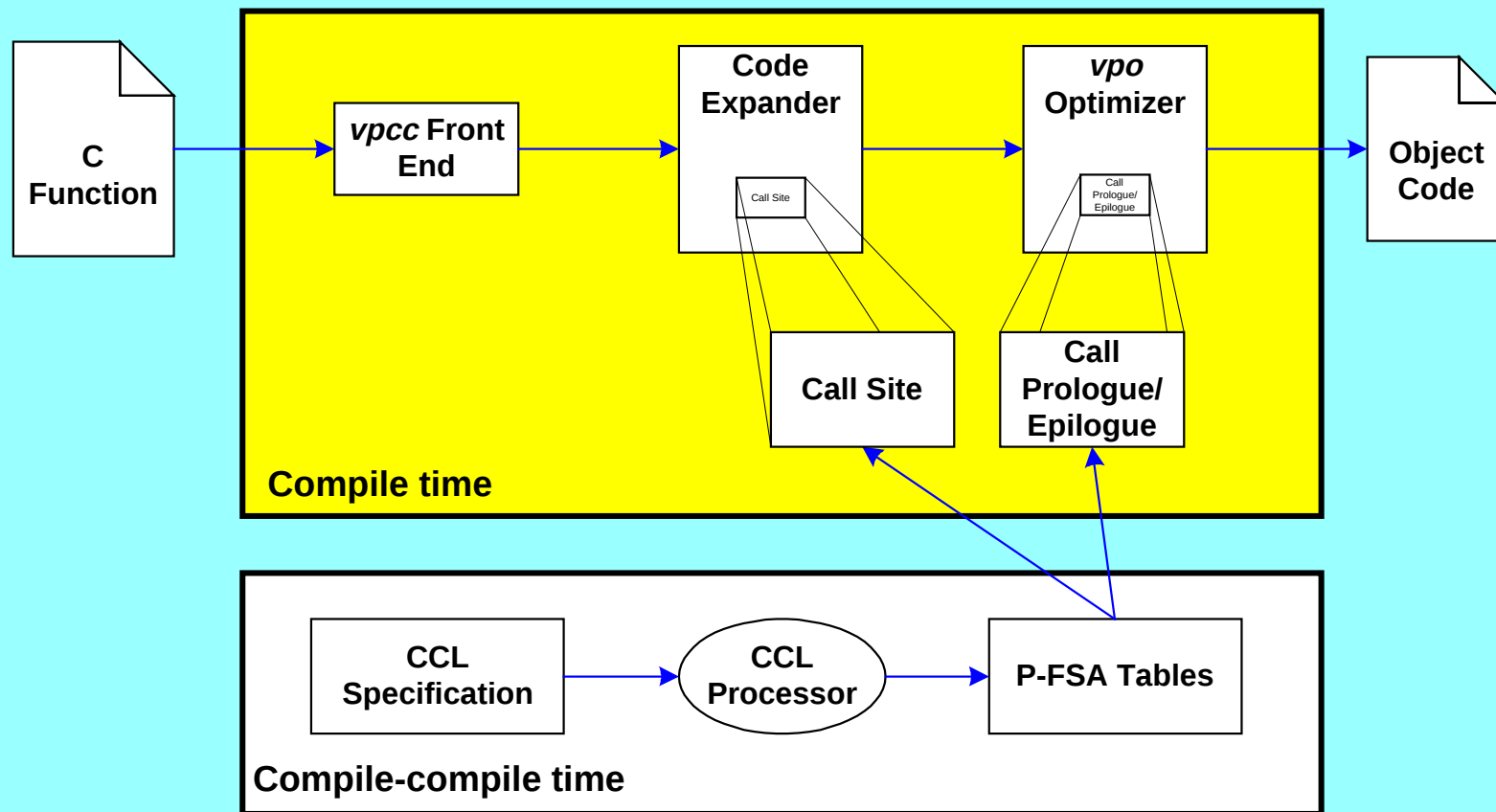
University of Virginia

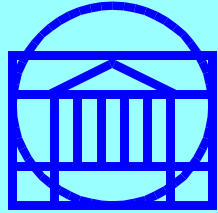




CCL - Calling Convention Language

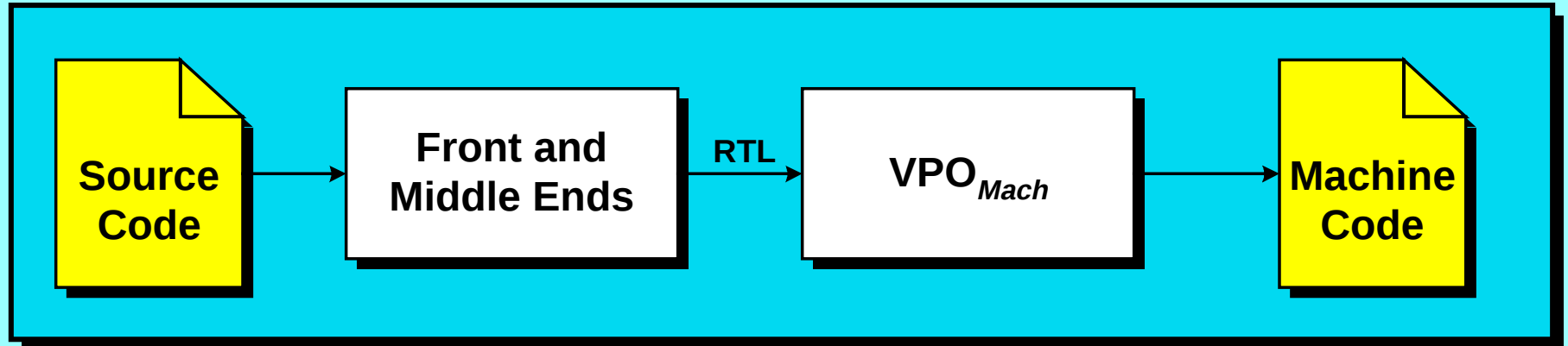
University of Virginia

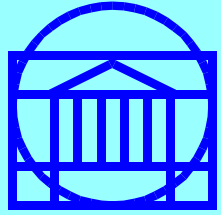




Compilation with VPO

University of Virginia

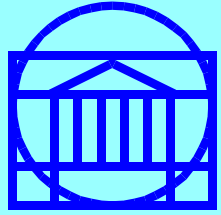




VPO Optimizations

University of Virginia

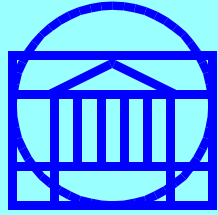
- ◆ Register allocation
- ◆ Loop-invariant code motion
- ◆ Evaluation order determination
- ◆ Constant propagation
- ◆ Loop unrolling
- ◆ Inline function expansion
- ◆ Constant folding
- ◆ Common subexpression elimination
- ◆ Induction variable elimination
- ◆ Dead code elimination
- ◆ Instruction scheduling
- ◆ Memory access coalescing
- ◆ Recurrence detection and optimization



VPO Design Principles

University of Virginia

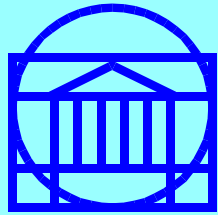
- ◆ All code improvements are effectively machine dependent
- ◆ Instruction selection must be available “on demand”
- ◆ All code improvements are important some of the time



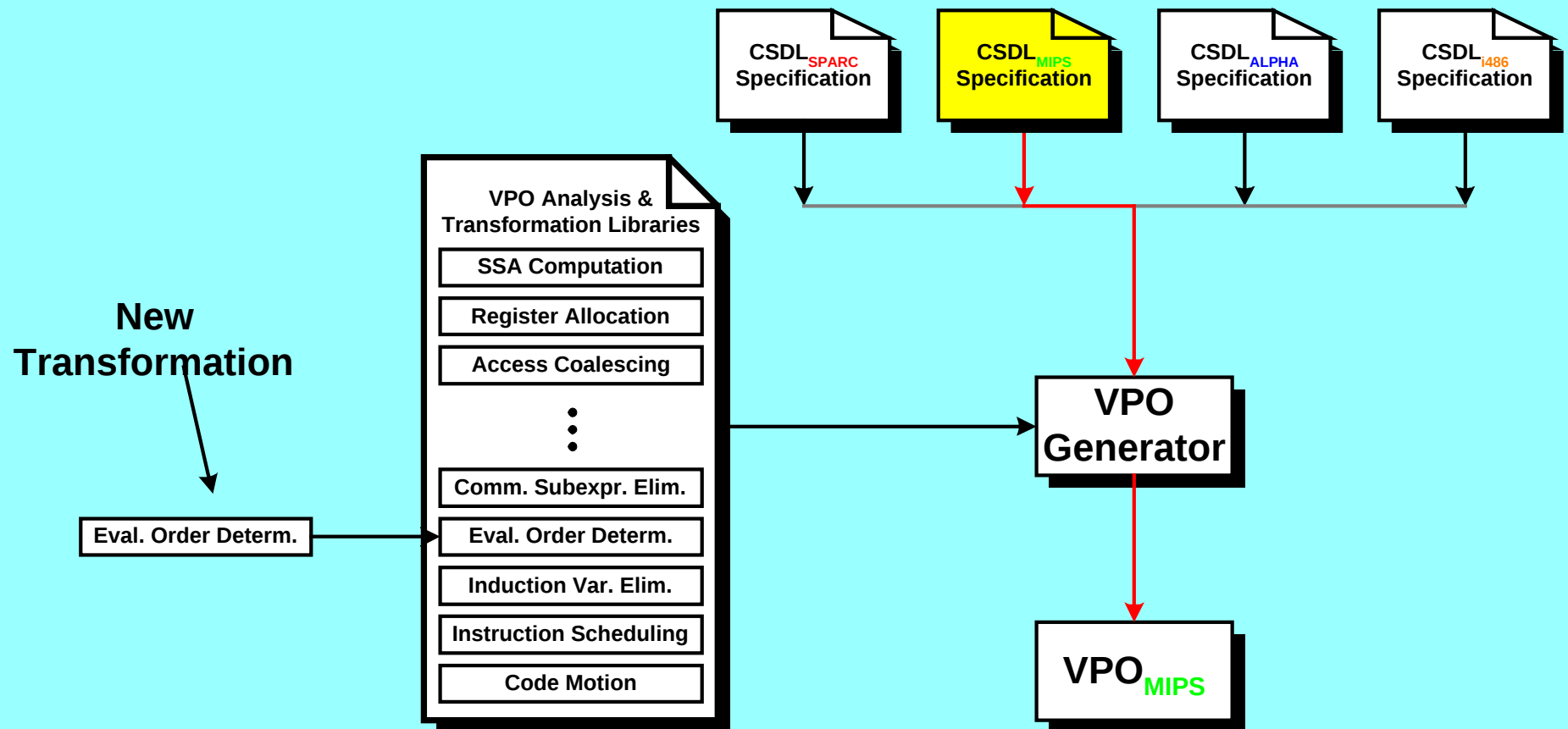
VPO Architecture

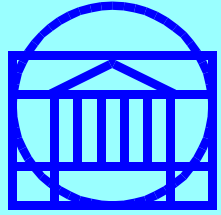
University of Virginia

- ◆ **Optimizes a single, low-level program representation (RTL)**
 - ◆ Exposes target machine dependencies
 - ◆ Allows optimizations to be applied in any order and repeated
 - Simplifies implementation of optimizations
 - Reduces phase-ordering problems
- ◆ **Retargeted by supplying a description of the target system (CSDL)**
- ◆ **Uses a retargetable peephole optimizer to perform instruction selection on demand.**



Building a VPO Optimizer

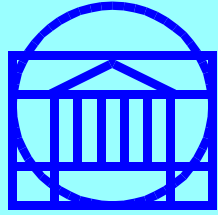




VPO Features

University of Virginia

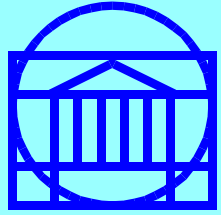
- ◆ **Front end and language independent**
 - ◆ C/C++
 - ◆ Ada
 - ◆ PL/I
 - ◆ Pascal
- ◆ **Optimizations are “plug-and-play”**
 - ◆ Easy to add new ones
 - ◆ Provides level playing field for evaluating different algorithms



VPO Features

University of Virginia

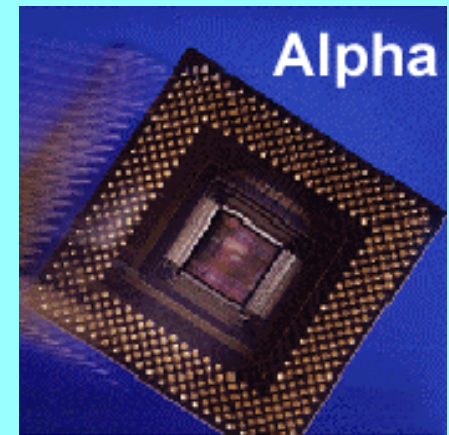
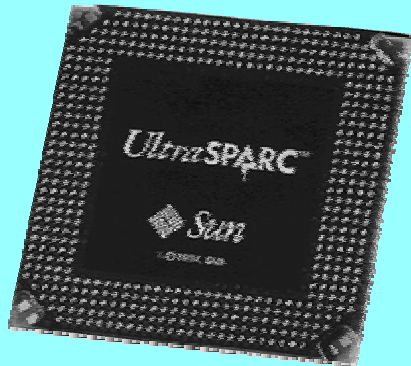
- ◆ **Small**
 - ◆ 30,000 lines of machine-independent code
 - ◆ 1,000 lines of machine-dependent code
- ◆ **Simple algorithms**
- ◆ **Small + Simple = Flexible**



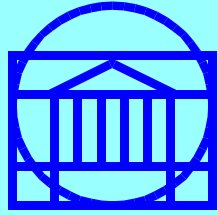
VPO Strengths

University of Virginia

- ◆ Easily retargeted (has been targeted to over 15 architectures)
 - ◆ Stock microprocessors



- ◆ DSPs
- ◆ Microcontrollers
- ◆ Media processors



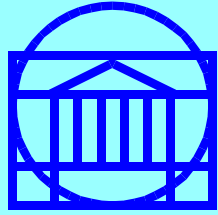
VPO Strengths

University of Virginia

◆ **Robust**

- ◆ **Small + Simple = Robust**
- ◆ **Industrial users**
 - Avionics
 - Consumer electronics
 - Control systems

◆ **Produces high-quality code**



Zephyr Time Table

University of Virginia

◆ **Sept '97**

◆ **ASDL**

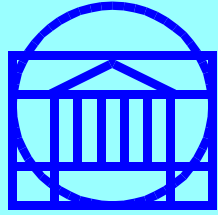
- Preliminary design and implementation available for users and comment
- SUIF description available

◆ **CSDL**

- Annotated CSDL descriptions for the SPARC, 80X86, and Alpha ISA core available for public comment and feedback

◆ **VPO**

- VPO connected to lcc released (SPARC)
- Interface to VPO documented
- Retargeting documented



Zephyr Time Table

◆ Jan '98

◆ ASDL

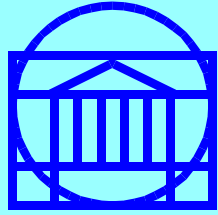
- Specification of lcc intermediate language complete
- Picklers and browsers available

◆ CSDL

- CSDL pipeline specification started
- Core CSDL specification and semantics complete

◆ VPO

- Conversion to Zephyr intermediate form (ZIF) complete
- Additional lcc/vpo target released (DEC Alpha)

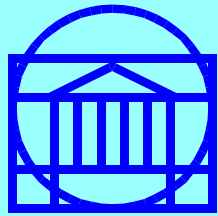


Zephyr team

University of Virginia

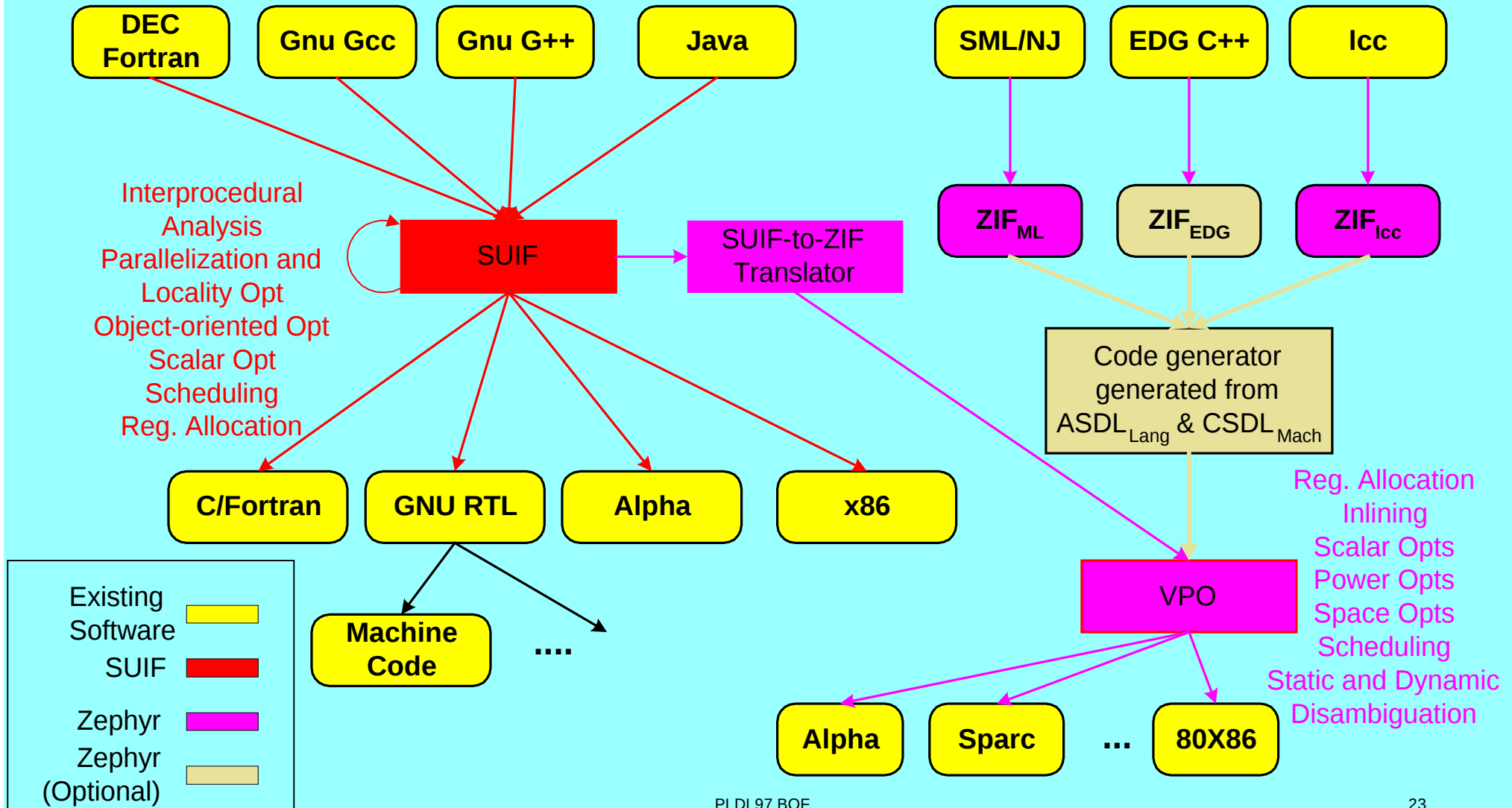
- ◆ **Andrew Appel - Princeton (SML/NJ)**
- ◆ **Jack Davidson - Virginia (vpo, ANDF)**
- ◆ **Norman Ramsey - Virginia (NJ Toolkit)**
- ◆ **Bill Wulf - Virginia (Bliss-11, PQCC, Tartan)**

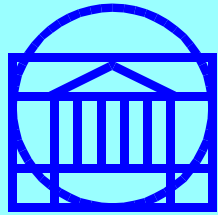
<http://www.cs.virginia.edu/nci>



SUIF and Zephyr

University of Virginia





Future Directions

University of Virginia

- ◆ Automate all aspects of compiler and system software construction

