Title:	LLVM Infrastructure and Tools Project Summary
Author(s):	McCormick, Patrick Sean
Intended for:	ECP/ATDM project documentation.
Issued:	2017-11-06

2.3.2.01 – LLVM Infrastructure and Tools

This project works with the open source LLVM Compiler Infrastructure (http://llvm.org) to provide tools and capabilities that address needs and challenges faced by ECP community (applications, libraries, and other components of the software stack). Our focus is on providing a more productive development environment that enables (i) improved compilation times and code generation for parallelism, (ii) additional features/capabilities within the design and implementations of LLVM components for improved platform/performance portability and (iii) improved aspects related to composition of the underlying implementation details of the programming environment, capturing resource utilization, overheads, etc. -- including runtime systems that are often not easily addressed by application and library developers. Current efforts are focused on the needs of the FleCSI framework (1.3.3.12) and La Ristra application (1.2.2.01) projects. Many aspects of this work include applying our techniques to codes that use Kokkos (1.3.1.05) and RAJA (1.3.1.08) – whom we collaborate with to understand the challenges they are facing with regards to the focus areas above. We focus on the C Family of languages (primarily C++) and are actively working with the Flang Project (**2.3.5.X -- not sure where this ended up**) to expand coverage to include Fortran. Additionally, there are synergies with the PROTEUS project (1.3.2.12) where collaborate on the use of higher-level forms of intermediate code representation with LLVM. Finally, we will actively engage the broader LLVM community to explore and initiate the steps needed to have our efforts incorporated into the infrastructure – our first target is participating in the parallel intermediate representation working group that has recently formed. This step is a critical aspect to achieving a long-term (post ECP) solution for the DOE community and also in terms of making the LLVM infrastructure a better match for addressing ECP challenges.

LLVM Infrastructure and Tools Overview

Scope & Intent	R&D Themes	Delivery Process	Target ECP Users	Support Model
Research, design and development and support for an ECP-aware LLVM compiler infrastructure. Long-term contributions of our work back into the LLVM infrastructure for wider deployment.	Compiler infrastructure and supporting implementation details including developer productivity and performance portability aspects.	Regular open-source releases of software on GitHub that follow the established phases of the LLVM Project as well as releases that address reported issues and feature requests from the ECP community.	Applications and/or libraries using the C, C++ or Fortran families of languages. We are especially interested in those that are having issues with efficient/optimized code generation, significant compile-time overheads, and with platform and performance portability concerns.	Ongoing developer support. Dedicated email and github issue tracking, and open source access. Work with LLVM community for adoption of our contributions.

LLVM Infrastructure and Tools FY18 Milestones

Note this project is part of the NNSA ASC Co-Design L2 Milestone for FY18.

Milestone ID	Milestone Title	ECP Users
NA	Verison drop of custom LLVM infrastructure exploring improved complication of OpenMP (C++ and Fortran), FleCSI, Kokkos and RAJA constructs.	La Ristra (1.2.2.01), FleCSI and codes that utilize OpenMP (C++ or Fortran/Flang), Kokkos and/or RAJA.
NA	Verison drop of LLVM infrastructure utilizing improved intermediate form for representing, analyzing and optimizing parallel constructs.	La Ristra (1.2.2.01), FleCSI and codes that utilize OpenMP (C++ or Fortran/Flang), Kokkos and/or RAJA.
NA	NNSA ASC Co-Design L2 final report	NNSA ASC, ECP leadership, ECP projects.

Impact goals and metrics: List 2 – 3 impact goals and how you will measure progress.

LLVM Infrastructure and Tools Impact Goals & Metrics

Goal	Metric
Improve compile times for complex C++ code constructs focused on those used in FleCSI, Kokkos and RAJA. We also hope to explore related aspects here such as intermediate and executable file sizes.	Provide a modified implementation of the LLVM compiler infrastructure that provides improved compile times and lower overheads (e.g. reduce intermediate object file and executables sizes) for C++ codes that have known (well defined) syntax and parallel semantics.
Improve LLVM's ability to handle (analyze, optimize) parallel code constructs via the use of a high-level intermediate representation. At the same time engage with the LLVM community to help push for this capability in the standard LLVM implementation. Aspects here will range from OpenMP, FleCSI, Kokkos and RAJA.	Show improved anlaysis and/or optimization of C++/C and Fortran codes with known (well defined) parallel semantics. As applicable we will expand our experiments here to include aspects of production codes within ASC.