# 実装言語独立でモジュラリティーの良いコンパイラキット SCK

## 1　背景

コンパイラキットはユーザにとっての使いやすさが重要である．コンパイラの扱うデータは複雑であるから，ソースが公開されていて，ドキュメントが整備されているだけでは十分とは言えない．

例えば，現在最も広く使われているリターゲッタブルコンパイラである GCC は C 言語で実装された優れたオープンソフトであり，中間言語やリターゲッタブルコード生成 系のドキュメントも整っている強力なコンパイラであるが，いざ内部をいじろうとすると，複雑なデータ構造，グローバル変数の使用，メモリー管理の危険性などが障害となり，それは容易ではない．本来，GCC は強力で実用的なフリーのリターゲッタブルコンパイラを作成するのが目的であったから，コンパイラキットとしての視点，すなわち各部品のモジュラリティーの良さや簡潔さは考慮されていない．

## 2　目的

もっとコンパイラ研究者にとって使いやすいコンパイラインフラを作ろうという反省から，COINS (COmpiler INfraStructure) プロジェクトが発足した．開発代表者も COINS プロジェクトに参加していた．COINS コンパイラはその点実装言語が Java になったことから改善されている．C 言語で実装されている gcc における複雑で危険なメモリー管理からは開放され，C 言語特有のトリッキーなポインタ操作も Java のオブジェクトレベルでの安全な操作で可能になっている．

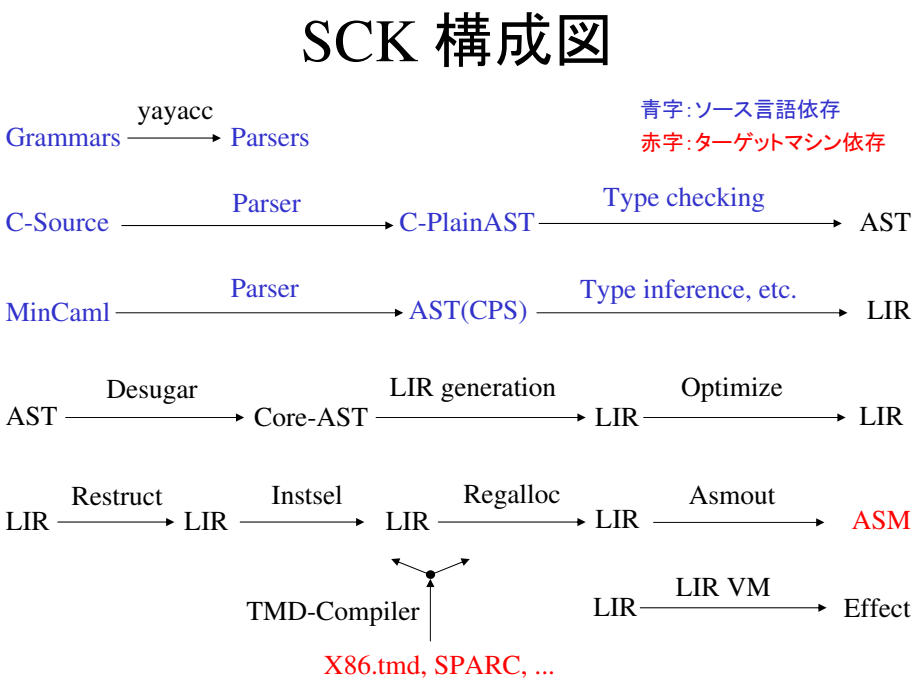
一方で，我々は依然としてこれらのコンパイラキット (インフラ) には大きな問題点があると考える．それは我々のプロジェクトのタイトルにもある「実装言語からの独立性」である．モジュラリティーはどのようなソフトウエアでも重要であるが，特にソフトウエァの個々の構成部品を部分的に利用したり，あるいは差し替えたりする必要性があるコンパイラキットでは，モジュラリティーこそ使いやすさの最重要ポイントである．

## 3　開発の内容

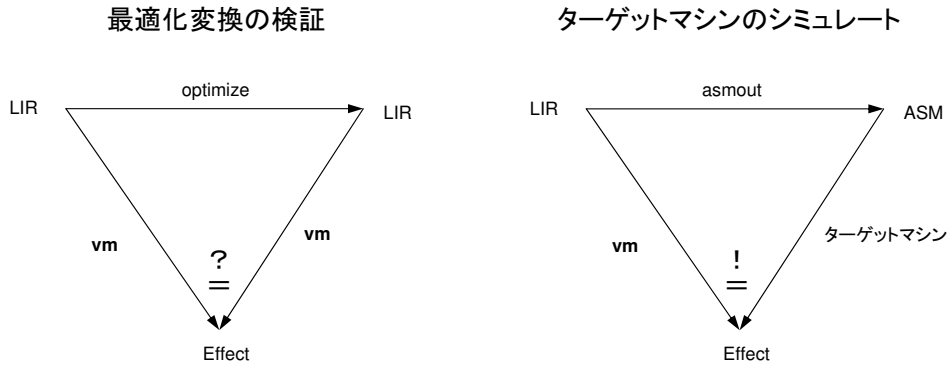
昨年度の未踏プロジェクト「実装言語独立でモジュラリティーの良いコンパイラキット」として作成したマルチソース・マルチターゲットのコンパイラ作成支援環境 SCK (S-expression based Compiler Kit) の上で，コード最適化等，コンパイラ開発支援環境として重要なモジュールを開発した．一方で，短い開発期間で作成された SCK はコード最適化，ドキュメンテーション，サポートしているソース言語とターゲット (現在は C 言語， X86 と SPARC のみ) の点でまだ不十分であった．そこで，今回の未踏プロジェクトにおいては以下の開発を行った．

1. yayacc への完全自動エラーリカバリ機能の導入
2. もう一つ（C 言語以外）の入力言語のサポート
3. もう一つのターゲットマシンのサポート
4. コード最適化モジュールの充実
5. ユーザインタフェースの充実

コード最適化において可能な限り我々のスタイルを踏襲し，モジュラリティーの高いコンパイラ部品を作成した．また，SCK プロジェクトの一部として作成されたパーサジェネレータ yayacc にも，フルオートマティックなエラーリカバリを導入すること (現在そのようなパーサジェネレータで広く使われているものは存在しない) によって，SCK のマルチソース対応をより洗練されたものになった．




また，LIR(SCK の中間言語) インタプリタを作成し，ターゲットマシン上の実行を Emacs 上で忠実にシミュレートすることも可能となった．これは LIR を忠実にシミュレートするインタプリタであり，デバッガ機能を有する．すなわちステップ実行したり，途中で変数の値を参照設定したりすることが可能である．




この仮想マシンの応用としては， コード最適化の正しさの検証への利用（左）の他に，ターゲットマシン記述に従って命令選択とレジスタアロケーションを経て，最終的なアセンブラ出力の一歩手前の LIR を実行する（右）という利用が考えられる．つまりこの場合，LIR の各命令ははセマンティカルにはターゲットマシンの機械語に対応しているので，この実行はターゲットマシンの命令を実行することに相当する．

## 4　従来の技術との相違

SCK の特徴は大きく二つに分けられる．それはモジュラリティーの良さと開発環境の良さである．前者はコンパイラの部品を実装言語と独立な S-式ベースの単純なインタフェースを持つモジュールに分割することで，コンパイラ開発者，研究者にとって使いやすい部品を提供している．以下に主要なモジュールのモジュラリティー (独立性) をまとめておく．

| | 実装言語独立 | C 言語独立 | LIR 独立 |
|---|---|---|---|
| yayacc | Yes | No | Yes |
| cpps | Yes | No | Yes |
| cps | Yes | No | Yes |
| reduce | Yes | yes | Yes |
| tmd | No | Yes | No |
| loop | Yes | Yes | Yes |
| live | Yes | Yes | Yes |
| regalloc | Yes | Yes | Yes |
| optimizers | Yes | Yes | No |

一方，後者は Emacs Lisp という実装言語により SCK を実装したことにより，Lisp のもつ強力さ，ラピッドプロトタイピングのしやすさに加え，Emacs のインタラクティブな環境を利用出来るという点である．

## 5　期待される効果

Emacs のエディタコマンドとして手軽に個々の部品を実行し，テストすることが出来るという特徴は他に類を見ない．これは新たなコンパイラを開発する場合にも威力を発揮するし，また個々のコンパイルフェーズが行う処理を理解することも容易になるためコンパイラ教育の教材としても優れている．付属の　graphviz へのインタフェースによる可視化ツールは，デバッグのみならず，コンパイラの教材を作る上でも，また，コンパイラの授業をアクティブで楽しいものにする上でも，重要な価値がある．

## 6　普及の見通し

生まれて間もない SCK ではあるが，デモを見た人は例外なく「面白い」という感想を漏らしてくれる．首都大学東京で SCK を教材としてコンパイラの授業を行ったが，学生からはゲームのようで面白いという感想が得られた．現時点では主に教育目的で教材としての利用を第一に考えている．

## 7　開発者名

- 阿部 正佳 （フリープログラマ）
- 山崎 淳 （株式会社ミラクルアーツ）
- 山根 雅司 （株式会社ワイズケイ）
- プロジェクトホームページ
  `http://www14.plala.or.jp/gazico/sck/`