コンパイラ資料(コード生成概要)　担当：佐々木晃

コード生成（「算術式」のコンパイル（翻訳）の例）

コンパイラにおけるコード生成フェーズとは、それまでの解析によって得られた結果を使って翻訳結果を出力するフェーズである。コード生成の例としては、実はすでに出てきた。
「逆ポーランド記法へ」の「翻訳」→前回のサンプルプログラム

演習 1：次のようなメッセージを出すようにせよ。（逆ポーランド記法への変換プログラムの出力部分を変えればよい。）(1)の例→ex9¥*.java
http://cis.k.hosei.ac.jp/~asasaki/lect/compiler/2008B/ex9-javaccSamples.zip

(1)日本語による命令列(Instruction Sequence)に翻訳

オペレータの場合：
「□２つの数 x, y をスタックからおろして足します。
そして、その結果をスタックトップに乗せます（ロードします)。」
オペランドの場合：
「□５をスタックトップに乗せます(５をロードします)。」

(2)英語による命令に翻訳

□Get two numbers x, y from the stack top, add y and x,
and load its result to the stack top.
□Load Constant 5 to the stack top.

→注意、「翻訳」と書いたが、実際にはこのメッセージを読む人がスタックというものを知っていることが前提となっている。

仮想機械hsm（スタックマシン）

これから作成しようとするコンパイラの目的機械。
http://cis.k.hosei.ac.jp/~nakata/lectureCompiler/HiStackMachine-ALL/index.html

演習 2：hsm の命令に翻訳せよ。
(1) 3 + 5 * 1
(2) 3 * (5 - 1)

演習 3：演習 1 で出力するメッセージを変更して仮想マシン hsm の命令を生成できるようにせよ。

1. 「□直前の２つの数 x, y をスタックからおろして X ます。
そして、その結果をスタックトップに乗せます（ロードします)。」
は、下記の hsm の演算命令 AD, SB, ML, DV 命令に対応する。
X="足し" →AD 0 0,　X="引き" →SB 0 0, X="掛け" →ML 0 0, X="割り" →DV 0 0

2「□n をスタックに乗せます。(n をロードします)。」は hsm の「LDC 0 n」という命令に対応する。

即値（リテラル＝定数、コンスタント）のロード命令（LoaD Constant）

→ちなみに jvm の場合

演算命令は、iadd, isub, idiv, imul 、即値ロード命令 iconst n という命令になる。

演習 4: hsm で演習 2 や演習 3 で作成した目的コードを実際に実行してみよ。最後に停止命令「HL 0 0」を置かなければならないことに注意。Java/Swing 版の hsm を用いること↓
http://cis.k.hosei.ac.jp/~nakata/lectureCompiler/JavaHiStackMachine-ALL/index.html

演習 2(1)例
[LDC 0 3; LDC 0 5; LDC 0 1; ML 0 0; AD 0 0; HL 0 0]
(→ 3 5 1 * +)