COMPILER

2 PASS COMPILER

Thang PM.

CSIM / SET / AIT

March 25, 2014

In this lab, we will learn how to construct a two-pass compiler, which

- first pass: parses input strings to build their parse trees
- second pass: transforms these trees into numbers

The main extensions:

- CUP file contains the following lines
  1. to redefine the data types of non-terminals expr, term, factor:
     nonterminal Aexp expr, term, factor;
  2. to begin the second pass as soon as the whole input expression has been parsed:
     program ::= expr:e
     {: System.out.println(e.getexp() + "==>" + e.getval()); :};
  3. to return the value of expression e by method *e.getval()*

- Class Aexp represents arithmetic expressions with their explicit parse trees.
- Syntax-directed translation for building Aexp objects is implemented in CUP file as follows:

```
expr ::= expr:e PLUS term:t
{: RESULT = new Aexp(new Args(e,t), sym.PLUS); :}
```