Compilers
(Compiling Techniques)

Programs

Manipulating

Programs According to the Meaning

Programs manipulating Programs

What are meaning the following writings?

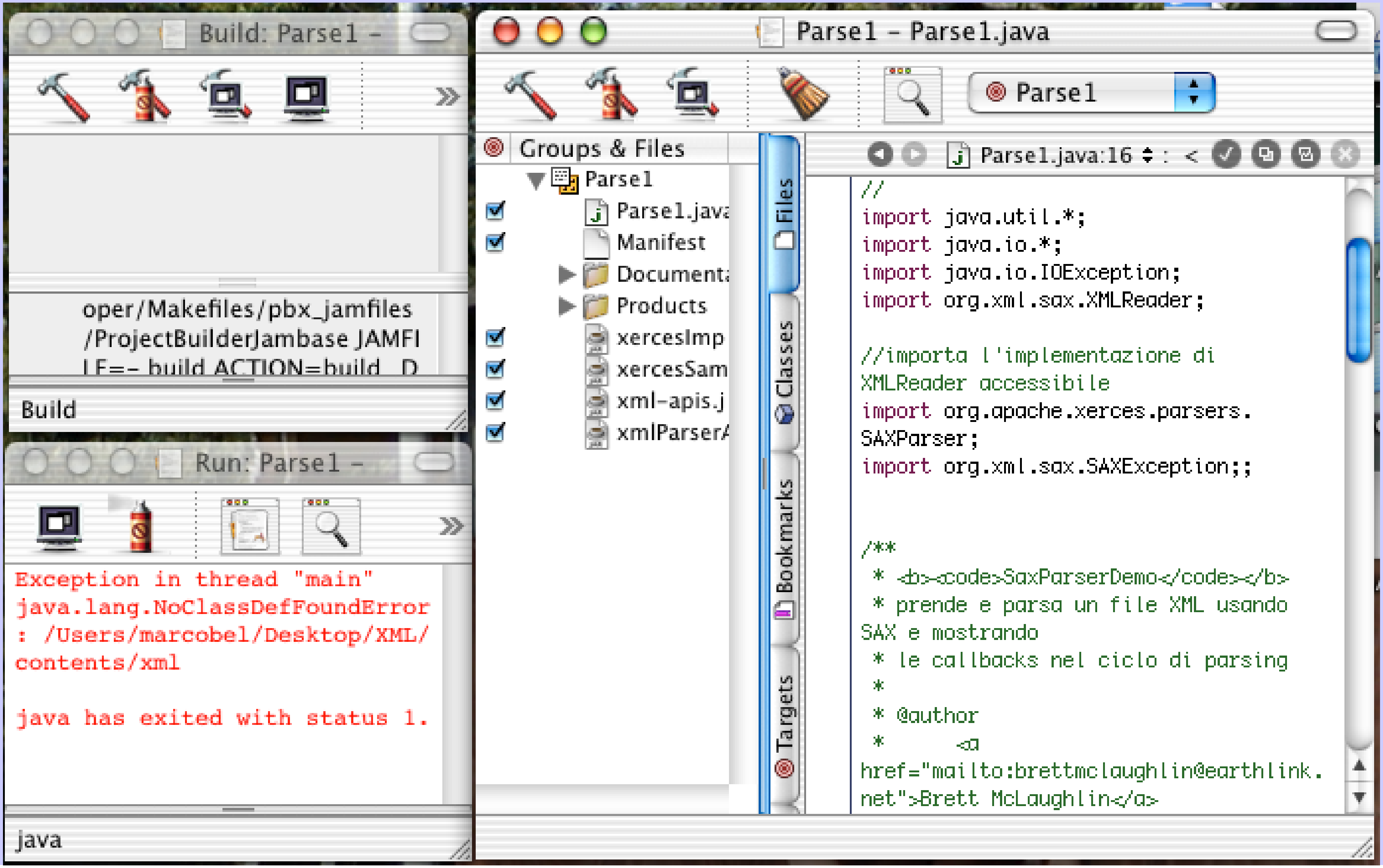
```
• cc -w -o  code.exe  ex.yy.c
• code.exe A.file B.file  <In  >Out

• javac -classpath /docu/XML/Xerces/Xparse.jar -v MyDoc.java
• java MyDoc
```

- **What are** ex.yy.c,   MyDoc.java **?**
- **What are** cc, javac, java **?**

- cc, javac, java **are programs manipulating programs:**
  - **How are defined (built, obtained) ?**
  - **Where are running?**
  - **In what language are they written?**

Compiler I/0 may be inline or through a graphic interface



```
oper/Makefiles/pbx_jamfiles
/ProjectBuilderJambase JAMFI
LE=- build ACTION=build D
```

```
Exception in thread "main"
java.lang.NoClassDefFoundError
: /Users/marcobel/Desktop/XML/
contents/xml

java has exited with status 1.
```

```
//
import java.util.*;
import java.io.*;
import java.io.IOException;
import org.xml.sax.XMLReader;

//importa l'implementazione di
XMLReader accessibile
import org.apache.xerces.parsers.
SAXParser;
import org.xml.sax.SAXException;;


/**
 * <b><code>SaxParserDemo</code></b>
 * prende e parsa un file XML usando
SAX e mostrando
 * le callbacks nel ciclo di parsing
 *
 * @author
 *      <a
href="mailto:brettmclaughlin@earthlink.
net">Brett McLaughlin</a>
```

Many different I/0 interaction structures exist but they are unaffecting

- The (compiler) beaviour, hence:
  - its Construction Principles
  - its Construction Techniques
  - its Internal Structure

In this Course: The Techniques

Basic Techniques for the construction of **Tools for Abstract Machines**

Techniques: Automata (N/D Finite State, T/B Pushdown)
Syntax-Directed Translations
Attribute Grammars/Translation Scheme
Structure Traversal / Visit
Translation Invariants (code generation)
…

In This Course:
Methodologies, Tools

Methodologies: Semantic Attachment
Abstract Interpretation
Meta-evaluation
Partial Evaluation
…

Tools: Lexical/Syntactic Analysers
Semantic Analisers
Syntactic Editors/Text Formatters
(analysis) Tools Generators
Code Generators
Interpreters / Compilers
Debuggers
Code Optimization Tools…

Foundations

- Language and Abstract Machine (MA)
- MA: Structure and Computation (States)
- Construction of MAs: Interpreter, Compiler
- Interpreter: Inside
- Compiler: Run Time Support (RTS)
- Compiler: Development Machine,
  Hierarchy Source-Host-Target (SOT)
- Intermediate Machines: Mixed Constructions

Definitions:

Language and Formalism

Language (Programming L.) =

= Formalism that Rigorously Expresses
(applications of computable functions)

Formalism = **Syntax** (*form* of the allowed sentences)
+
Semantics (*meaning* of each sentence)

Example: Programming Language

L = **<S, SEM>** is an LP

1) $\forall$ P$\in$S, SEM(P) $\in$\{N$\rightarrow$ N\}

2) $\forall$ g $\in$\{N $\rightarrow$ N\}, $\exists$ P $\in$S, such that:
g(n) = SEM(P)(n) ($\forall$ n $\in$N)

[Let {N → N} be the set of the
Computable Functions]

Definitions:

Abstract Machine

MA =

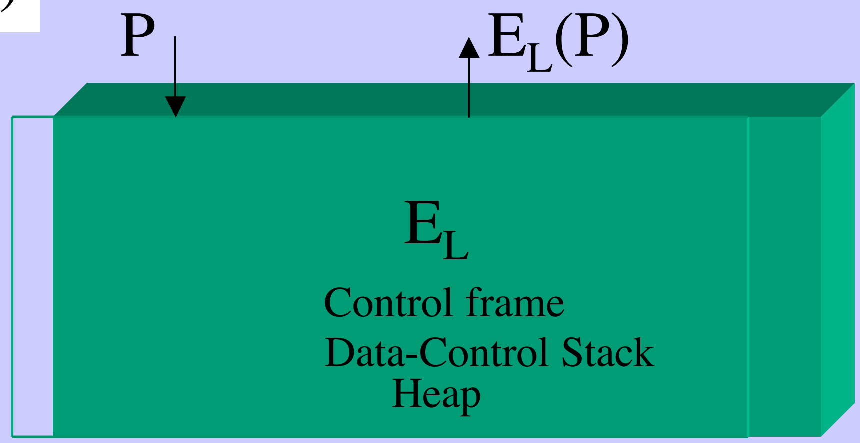
Machine Language (L=<S,SEM>)

+

Machine Executor (E_L)

$\forall\ P \in S,$

$$SEM(P) \approx E_L(P)$$



P

$E_L(P)$

E_L

Control frame

Data-Control Stack

Heap

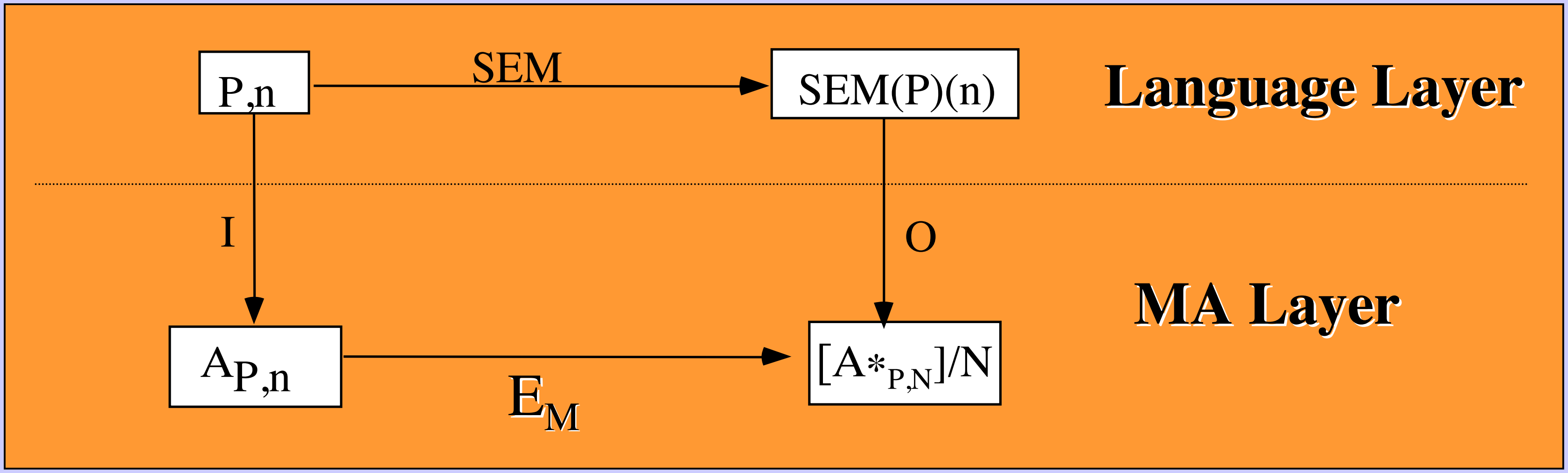
Java Virtual Machine

Landin’s SECD

Example: MA, Language and Machine Executor

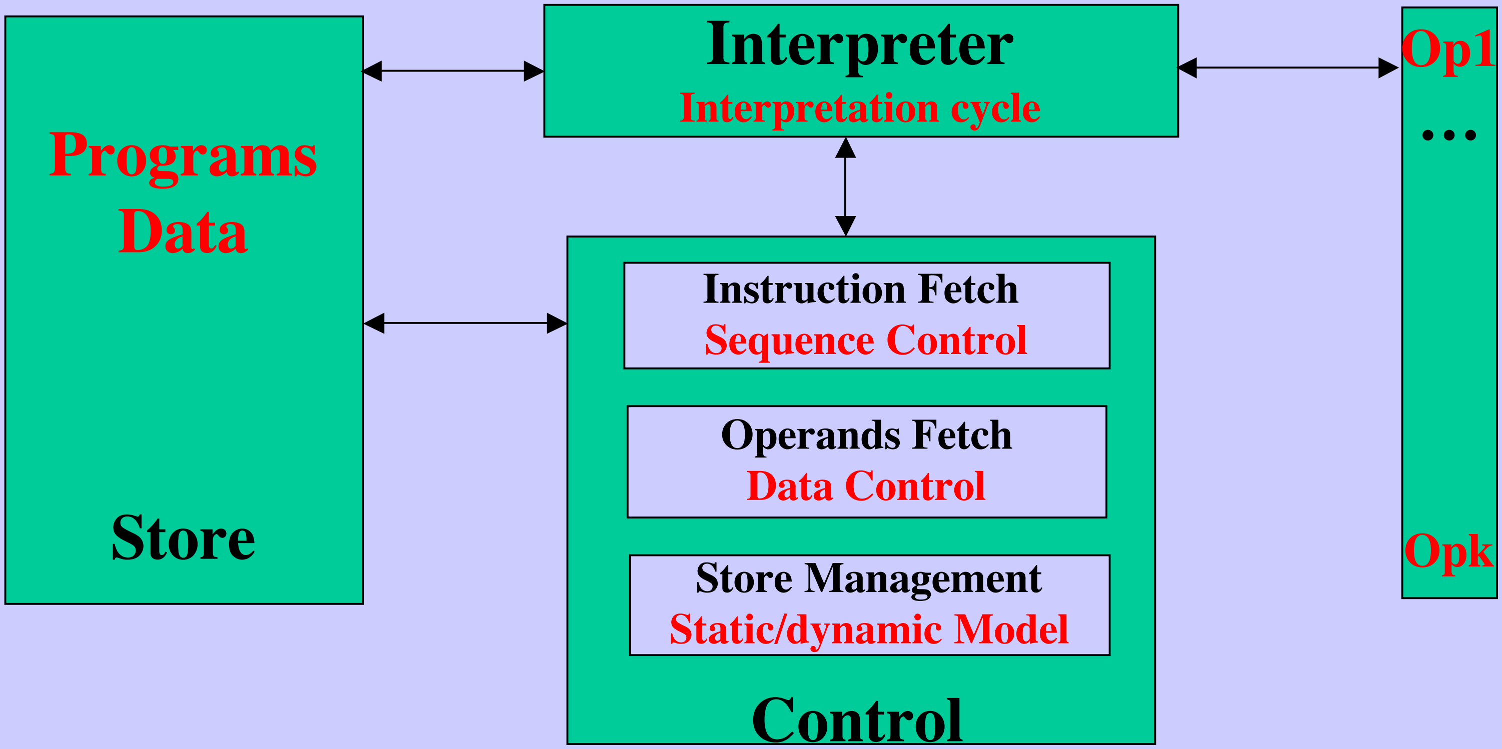
- L_M = <S, SEM> is an LP
- M = $<L_M, E_M>$
- E_M is the executor of M

$$E_M: A \rightarrow A^* \qquad \forall\, P \in S,\ n \in N$$



[where: I,O injections on A e A* risp., […]/N normalization on N]

MA: Structure e Executor States



Abstract Machine - Machine Structure

MA:
Store, Control

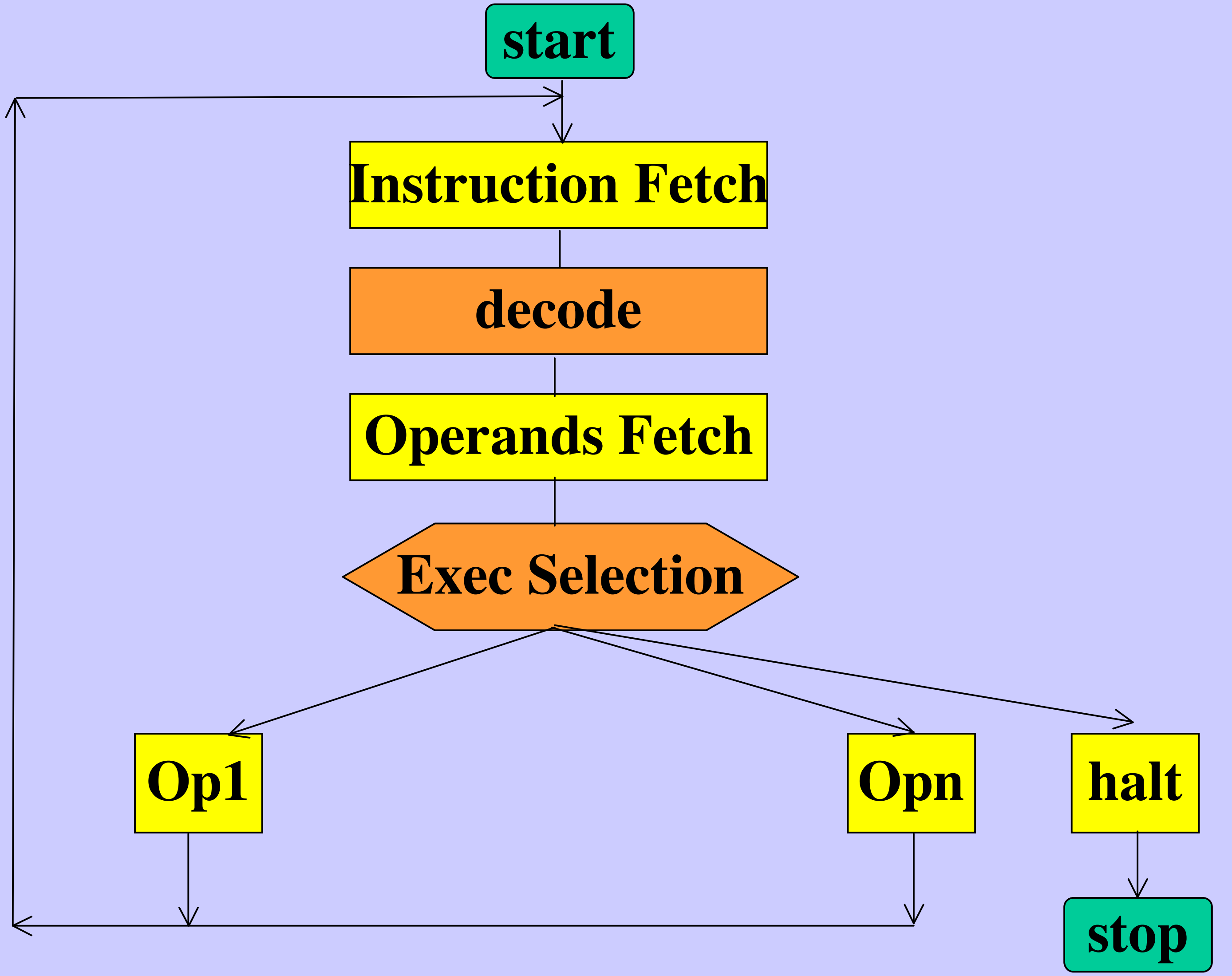
Store: **It is structured according to a model that relies on the specific features of the Machine Language**

- ***arrays of words, registers, stacks***
- **heap -** for dynamic allocation (Pascal, C, C++, … ,Java,
- ***graph -*** for structure sharing (*functional languages*)

Control: **It handles the Executor States:**

- **finds the next *statement* or expression**
- **finds the *stat*. or *espr*. data**
- **updates store**

MA:
Elementary Execution Cycle


start
Instruction Fetch
decode
Operands Fetch
Exec Selection
Op1
Opn
halt
stop