12048 Compilers II

Computer Science Engineering, 8º semester
Mandatory, 2º Cycle

Theoretical credits:	**3.0**
Lab credits:	**1.5**

Theory:
José Neira

Labs:
David Ken Vallejo,
Javier Fabra,
Luis Carlos Gallego

Schedule

Compilers II
Year 2007-2008

	Monday	Tuesday	Wednesday	Thursday	Friday
8-9		A			
9-10				A	
10-11					
11-12	C	C		Tutoring	
12-13				Tutoring	
13-14				Tutoring	
14-15					
15-16					
16-17		Tutoring			
17-18		Tutoring			
18-19		Tutoring		B	
19-20	B				
20-21					

12048, English group (C, 46)

- **Classes**
  - **Classes** will be taught in **English**
  - **Slides** are available in **Spanish** and **English**
  - Students can **ask questions** in **Spanish** or **English**
  - Students can **answer questions** in **Spanish** or **English**

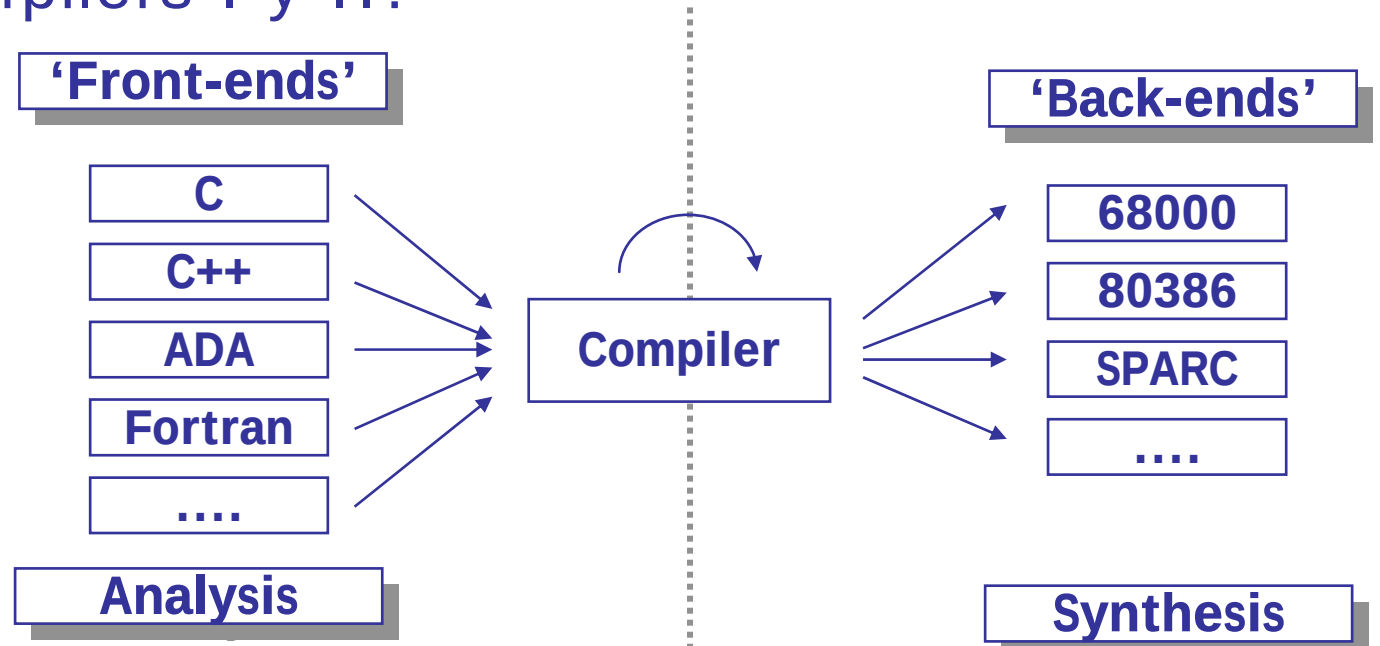
http://webdiis.unizar.es/~neira/

- **Lab work:**
  - **Lab** instructions will be in **Spanish**
  - **Reviewing** will be in **Spanish**

- **Exams:**
  - **Exams** will be in **Spanish**
  - Students can answer in **Spanish**

Preliminaries

- Compilers I y II:



Compilers I: theory

- Lexical analysis
- Syntactic analysis

Compilers II: pragmatics

- Semantic analysis
- Code generation
- Optimization

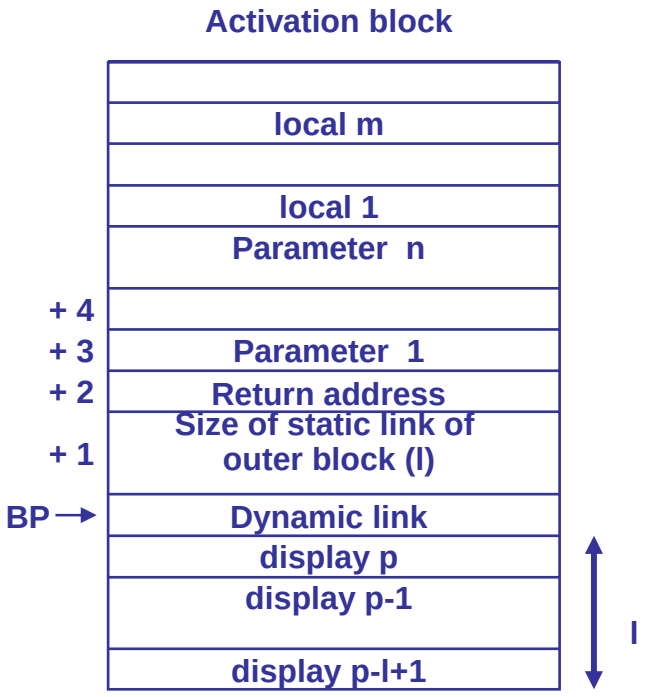
Preliminaries

- Prerequisites:
  - **12025 Languages, Grammars y Automata**
  - **12044 Compilers I**
- Grammars:

- What should I know beforehand?
  - **Lexical analysis using lex**
  - **Syntax analysis using yacc**

Preliminaries

- Prerequisites:
  - 12015 Computer Architecture



- What should I know beforehand?
  - Elements of an architecture
  - Instruction set
  - Addressing modes

```
STC n      push(n)
SRF n1 n2  push(display[DP - n1] + n2)
DRF        push (frames[pop()])
ASG        frames[pop2()] = pop1()
ASGI       frames[pop1()] = pop2()
```

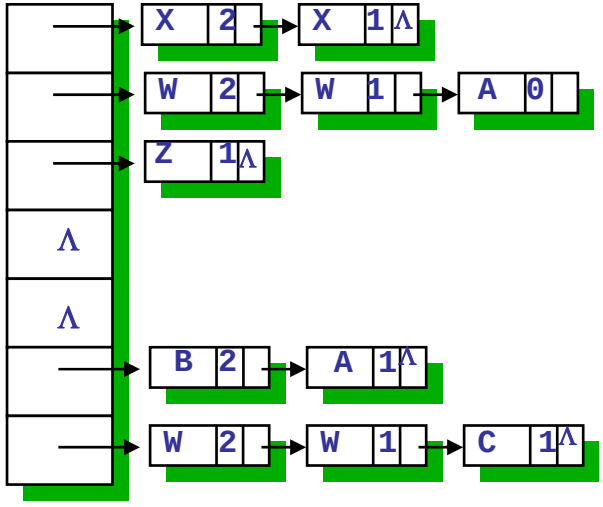
Preliminaries

- Prerequisites:
  - 12021 Algorithms and data structures:

Symbol tables

```
program main {
 int W;
 int X;
 void A() {
  int W;
  int Y;
  int Z;
  void B() { int X ; ... }
  void C() { int X; int Y ;...}
  ...
 } - end of A
 void D() { int Z ; ... }
}
```

- What should I know beforehand?
  - Tree structures
  - Search algorithms
    - » Hashing, ...



Preliminaries

- What can I learn?
  - **Semantic analysis** using yacc (use of attributes)
  - **Code generation** for stack machines
  - General techniques for **code optimization**

- What is left out?
  - **Interpreters**
    - » Other execution modes
  - **Debuggers**
    - » Increase efficiency in use of space
    - » Increase efficiency in execution time
  - **Unassemblers**
    - » Software recovery

Preliminaries: program

- **Introduction**
- **Part 1: Semantic analysis**
  - Lesson 1: Symbol tables
  - Lesson 2: Type checking
  - Lesson 3: Semantic analysis using attributed grammars
- **Part 2: Code generation**
  - Lesson 4: Abstract machines
  - Lesson 5: Generation of intermediate code
  - Lesson 6: Code optimization
- Exercises and problems in each lesson

Lab work

- Practical credits: 1.5

5 sessions 3 hour sessions in merlin (L0.04, Building A)

		Compiladores II Curso 2007-2008			
	Lunes	Martes	Miércoles	Jueves	Viernes
8-9					
9-10					
10-11					
11-12	Pr 3,6		Pr 1,2		
12-13	Pr 3,6		Pr 1,2		
13-14	Pr 3,6		Pr 1,2		
14-15					
15-16	Pr 4,5			Pr 7,8	
16-17	Pr 4,5			Pr 7,8	
17-18	Pr 4,5			Pr 7,8	
18-19					
19-20					
20-21					

Lab work

- Teachers

- Javier Fabra
  - Acerete to Dominguez

- Luis Carlos Gallego
  - Elbal to Muro

- David Ken Vallejo
  - Noé to Viñuales

Lab work

- What is done in lab work in Compilers **II?**
  - A Pascual Compiler to P(ortable) Code
  - A P Code assemblers
  - A P Code interpreter

- **We have chosen Pascal even if:**

  Why Pascal is Not My Favorite Programming Language
  Brian W. Kernighan, AT&T Bell Laboratories, 1981

  - It is a reasonably simple but complete subset of Pascal

- **We have chosen the P machine because:**

  Algorithms + Data Structures =
  Programs N. Wirth, P-H 1976

  - Code generation is very simple
  - Stack machines are in use at present

Lab work: comments

- Lab work is **individual**
- Learn from your colleagues, they are your best resource.
- You know perfectly well what constitutes copying
- Everything you submit must be **your own**
- Copying in lab work means grade 0 in lab work

Evaluation

- Final grade:

```
if (exam >= 5) and (lab_work >= 5) then
    grade := exam * 0.6 + lab_work * 0.4;
else
    grade := 'Suspenso';
end
```

- You must pass BOTH lab work and one exam

Evaluation

- **Lab work:**
  - Submit **the day specified in the instructions**
  - **We penalize** late lab work (one point per week, or fraction, maximum 5 weeks of delay).
  - If lab work is ok, the **minimum grade** will be **5.0**
  - Lab grades are **valid** for all exams
  - Lab work is **different** for each exam

- **Exam:**
  - Grade only valid for **one** exam
  - If you pass the exam but not the lab work, your grade will be **fail** (you will have to take another exam and submit new lab work).
  - **No** notes, books, exercises...

Bibliography

Compiladores: principios, técnicas y herramientas. A. Aho, R. Sethi, J. Ullman. Addison-Wesley Iberoamericana, 1993

The dragon book

Advanced Compiler Design and Implementation. S.S. Muchnick. Morgan Kaufmann Publishers, 1997

The whale book

Programming Language Pragmatics. M.L. Scott. Morgan Kaufmann Publishers, 2000

Engineering a Compiler. K.D. Cooper & L. Torczon. Morgan Kaufmann Publishers, 2004

State of the art

lex & yacc. J. Levine, T. Mason, D. Brown. O'Reilly & Associates, 1992

Reference manual

More information at

- Web page:
  http://www.cps.unizar.es/~neira/compii.htm
- Anillo Digital Docente:
  http://add.unizar.es
- Moodle:
  http://moodle.unizar.es