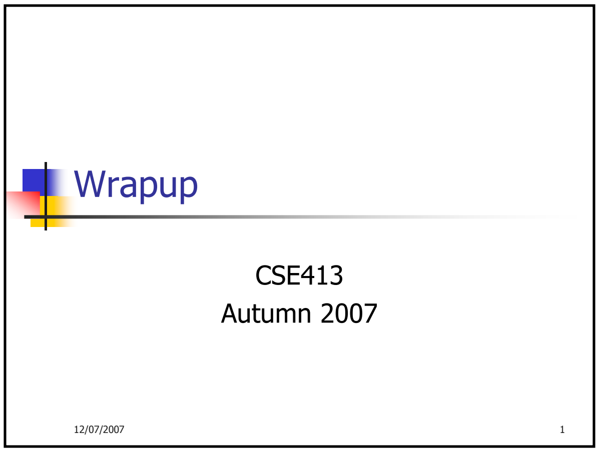
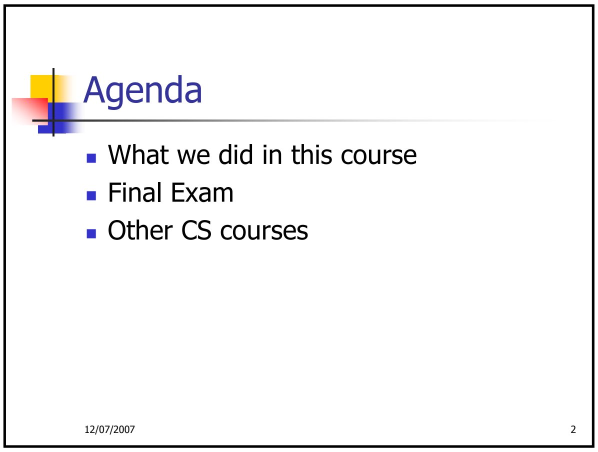


Wrapup

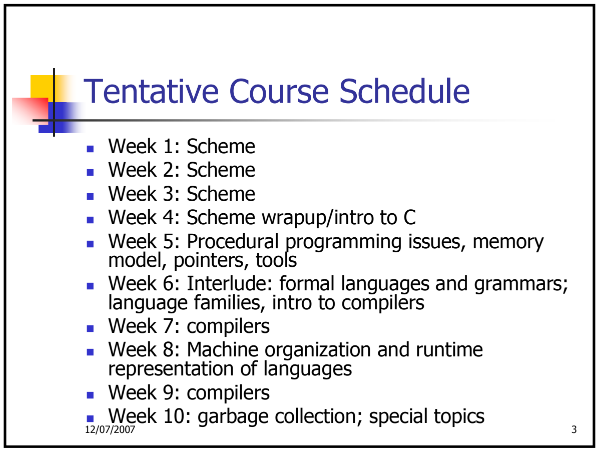
CSE413

Autumn 2007



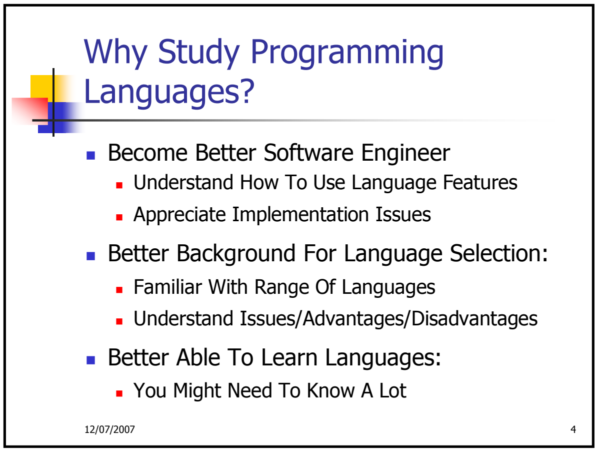
Agenda

- What we did in this course
- Final Exam
- Other CS courses



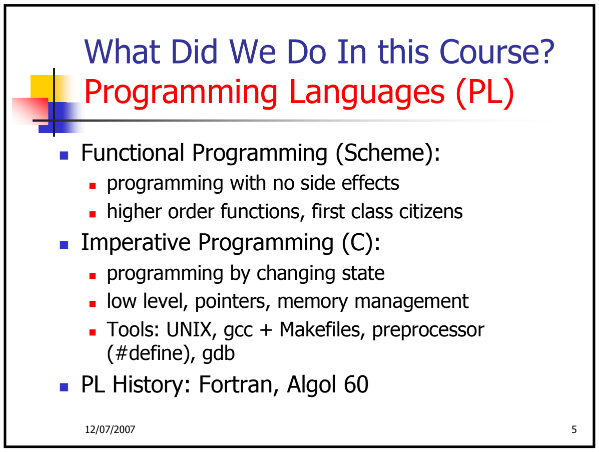
Tentative Course Schedule

- Week 1: Scheme
- Week 2: Scheme
- Week 3: Scheme
- Week 4: Scheme wrapup/intro to C
- Week 5: Procedural programming issues, memory model, pointers, tools
- Week 6: Interlude: formal languages and grammars; language families, intro to compilers
- Week 7: compilers
- Week 8: Machine organization and runtime representation of languages
- Week 9: compilers
- Week 10: garbage collection; special topics



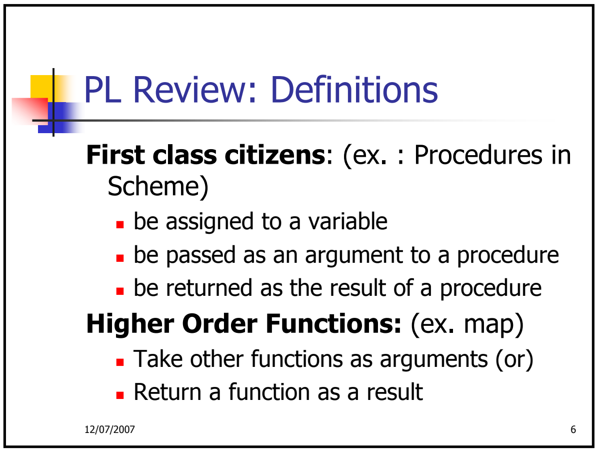
Why Study Programming Languages?

- Become Better Software Engineer
  - Understand How To Use Language Features
  - Appreciate Implementation Issues
- Better Background For Language Selection:
  - Familiar With Range Of Languages
  - Understand Issues/Advantages/Disadvantages
- Better Able To Learn Languages:
  - You Might Need To Know A Lot



What Did We Do In this Course? Programming Languages (PL)

- Functional Programming (Scheme):
  - programming with no side effects
  - higher order functions, first class citizens
- Imperative Programming (C):
  - programming by changing state
  - low level, pointers, memory management
  - Tools: UNIX, gcc + Makefiles, preprocessor (#define), gdb
- PL History: Fortran, Algol 60



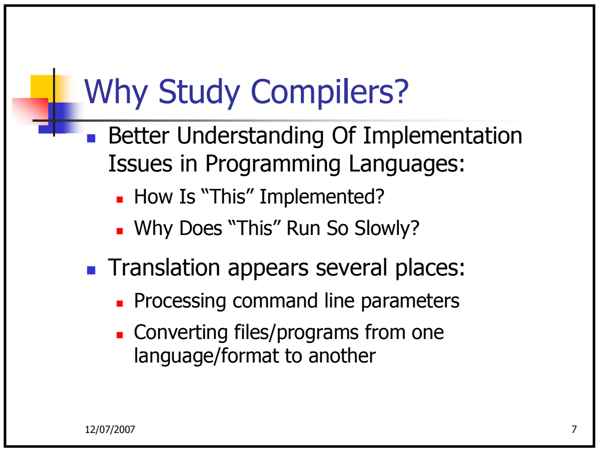
PL Review: Definitions

First class citizens: (ex. : Procedures in Scheme)

- be assigned to a variable
- be passed as an argument to a procedure
- be returned as the result of a procedure

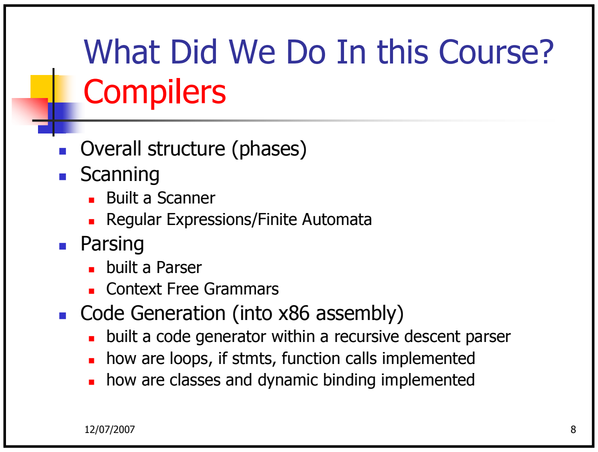
Higher Order Functions: (ex. map)

- Take other functions as arguments (or)
- Return a function as a result



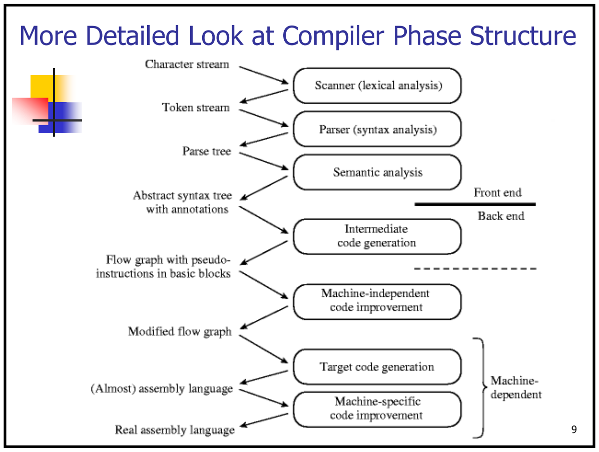
Why Study Compilers?

- Better Understanding Of Implementation Issues in Programming Languages:
  - How Is "This" Implemented?
  - Why Does "This" Run So Slowly?
- Translation appears several places:
  - Processing command line parameters
  - Converting files/programs from one language/format to another



What Did We Do In this Course? Compilers

- Overall structure (phases)
- Scanning
  - Built a Scanner
  - Regular Expressions/Finite Automata
- Parsing
  - built a Parser
  - Context Free Grammars
- Code Generation (into x86 assembly)
  - built a code generator within a recursive descent parser
  - how are loops, if stmts, function calls implemented
  - how are classes and dynamic binding implemented



More Detailed Look at Compiler Phase Structure

Character stream

Scanner (lexical analysis)

Token stream

Parser (syntax analysis)

Parse tree

Semantic analysis

Abstract syntax tree with annotations

Front end

Back end

Intermediate code generation

Flow graph with pseudo-instructions in basic blocks

Machine-independent code improvement

Modified flow graph

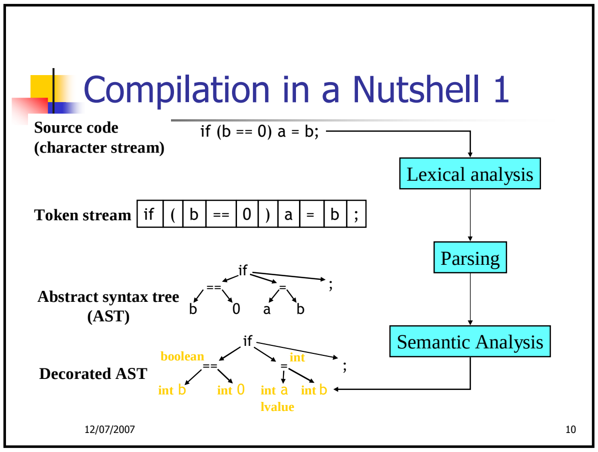
Target code generation

(Almost) assembly language

Machine-specific code improvement

Real assembly language

Machine-dependent



Compilation in a Nutshell 1

Source code (character stream)

if (b == 0) a = b;

Lexical analysis

Token stream

if	(	b	==	0	)	a	=	b	;

Parsing

Abstract syntax tree (AST)

if

==

b

0

=

a

b

;

Semantic Analysis

Decorated AST

if

boolean

==

int b

int 0

int

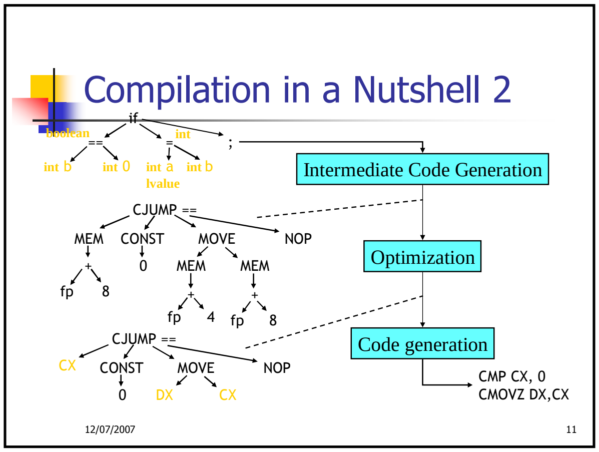
=

int a

lvalue

int b

;



Compilation in a Nutshell 2

if

boolean

==

int b

int 0

int

=

int a

lvalue

int b

;

Intermediate Code Generation

CJUMP ==

MEM

+

fp

8

CONST

0

MOVE

MEM

+

fp

4

MEM

+

fp

8

NOP

Optimization

CJUMP ==

CX

CONST

0

MOVE

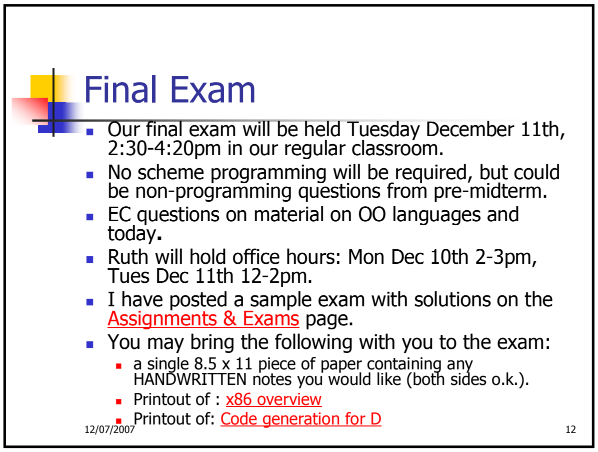
DX

CX

NOP

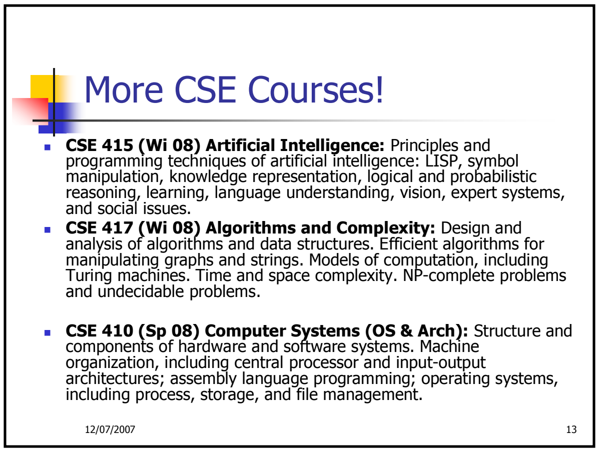
Code generation

```
CMP CX, 0
CMOVZ DX,CX
```



Final Exam

- Our final exam will be held Tuesday December 11th, 2:30-4:20pm in our regular classroom.
- No scheme programming will be required, but could be non-programming questions from pre-midterm.
- EC questions on material on OO languages and today.
- Ruth will hold office hours: Mon Dec 10th 2-3pm, Tues Dec 11th 12-2pm.
- I have posted a sample exam with solutions on the Assignments & Exams page.
- You may bring the following with you to the exam:
  - a single 8.5 x 11 piece of paper containing any HANDWRITTEN notes you would like (both sides o.k.).
  - Printout of : x86 overview
  - Printout of: Code generation for D



More CSE Courses!

- **CSE 415 (Wi 08) Artificial Intelligence:** Principles and programming techniques of artificial intelligence: LISP, symbol manipulation, knowledge representation, logical and probabilistic reasoning, learning, language understanding, vision, expert systems, and social issues.
- **CSE 417 (Wi 08) Algorithms and Complexity:** Design and analysis of algorithms and data structures. Efficient algorithms for manipulating graphs and strings. Models of computation, including Turing machines. Time and space complexity. NP-complete problems and undecidable problems.
- **CSE 410 (Sp 08) Computer Systems (OS & Arch):** Structure and components of hardware and software systems. Machine organization, including central processor and input-output architectures; assembly language programming; operating systems, including process, storage, and file management.

12/07/2007 13