University of Macau
Faculty of Science and Technology
Department of Computer and Information Science
CISB220 Compiler Construction
Syllabus
2nd Semester 2014/2015
Part A – Course Outline

Compulsory course in Computer Science

Course description:
(2-2) 3 credits. Modern compiler design, use of automatic tools, compilation techniques and program intermediate representations; scanner, recursive descent parser, bottom-up parser, code generation and optimization; semantic analysis and attribute grammars, transformational attribute grammars.

Course type:
Theoretical with substantial laboratory/practice content

Prerequisites:
- CISB111

Textbook(s) and other required material:
- David A Watt and Deryck F Brown. (2000) *Programming Language Processors in JAVA — Compilers and Interpreters*. Prentice Hall, US.

References:
- Alfred V Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D Ullman. (2007). *Compiler: Principles, Techniques and Tools*. 2nd Ed., Prentice Hall.
- Charles N. Fischer, Ron K. Cytron, and Richard J. LeBlanc. (2010). *Crafting A Compiler*. Pearson Higher Education.

Major prerequisites by topic:
- Programming languages and algorithms
- Logical operations
- Knowledge in trees and graphs
- Formal systems

Course objectives:
- Learn the operation of the major phases of a compiler
- Introduce the theories behind the various phases, including regular expressions, context free grammars, and finite state automata
- Apply the theoretical foundations that form the basis of compilation
- Design and implement parts of a compiler for a small imperative-style programming language
- Practice software engineering design principles on a medium size project

Topics covered:
- **Basic Concepts (4 hours)**: Review the concepts of high-level programming languages, and their syntax, contextual constraints and semantics, with examples from well-known programming systems. Introduce basic terminology of language processors: translators, compilers, interpreters, source and target languages, and real and abstract machines. Study the way of using language processors with Tombstone diagram.
- **Theoretical Foundations (4 hours)**: Review the fundamentals of formal language concepts, including finite-state automata, regular expressions, construction of equivalent deterministic finite-state automata from regular expressions, context-free grammars, grammar notation, derivations, and parse trees.
- **Syntactic Analysis (8 hours)**: Study the details of syntactic analysis, including scanning, parsing, and abstract syntax tree construction. Introduce recursive-descent parsing techniques, and show how a parser and scanner can be systematically constructed from the source language's syntactic specification.

- **Contextual Analysis (4 hours)**: Study the details of the contextual analysis module, in case of that the source language exhibits static bindings and is statically typed. Introduce the techniques to validate the identifier which relates to language's scope rules, and type checking which relates to the language's type rules.
- **Run-Time Organization (6 hours)**: Discuss the relationship between the source language and the target machine. Show how target machine instructions and storage must be marshaled to support the higher-level concepts of the source language. The topics include data representation, expression evaluation, storage allocation, routines and their arguments, garbage collection, and run-time organization of simple object-oriented languages.
- **Code Generation (6 hours)**: Study the details of code generation. Show how to organize the translation from source language to object code. It relates the selection of object code to the semantics of the source language in a stack-based machine. Basic techniques of code optimization are introduced at different phases: profiler optimization, intermediate code optimization and target code optimization.
- **Interpreters & Compiler Tools (3 hours)**: Look inside interpreters. It gives examples of interpreters for both low-level and high-level languages, as well as introduces the compiler construction tools: Lex & Yacc.

Class/laboratory schedule:

Timetabled work in hours per week			No of teaching weeks	Total hours	Total credits	No/Duration of exam papers
Lecture	**Tutorial**	**Practice**				
2	2	Nil	14	56	3	1 / 3 hours

Student study effort required:

Class contact:	
Lecture	28 hours
Tutorial	28 hours
Other study effort	
Self-study	28 hours
Homework assignment	6 hours
Project / Case study	15 hours
Total student study effort	105 hours

Student assessment:

Final assessment will be determined on the basis of:

Homework	10%	Project	20%
Mid-term	30%	Final exam	40%

Course assessment:

The assessment of course objectives will be determined on the basis of:

- Homework, project and exams
- Course evaluation

Course outline:

Weeks	Topic	Course work
1-2	**Introduction** Specification of programming language, language processors, Tombstone diagrams, bootstrapping, architecture of compiler, different analytical phases	
3	**Theoretical Foundations** Finite-state automata, regular expression, context-free grammar	
4-6	**Syntactic Analysis** Grammar transformation, parsing strategy, development of recursive-descent parser, intermediate representation (abstract syntax trees), scanner and error handling	Assignment#1 & #2 Project – Task#1
7-8	**Contextual Analysis** Organization of identification, type & scope checking, analysis	Project – Task#2

Weeks	Topic	Course work
	algorithm	
9-10	**Run-Time Organization** Data representation, expression evaluation, storage allocation, routines and heap storage allocation	Midterm exam Project – Task#3
11-12	**Code Generation** Code function, code template, generation algorithm, manipulation of constants & variables, procedures & functions	Assignment#3
13	**Interpretation** Interactive interpretation, recursive interpretation	
14	**Project Demonstration**	

Contribution of course to meet the professional component:
This course prepares students with fundamental knowledge and experiences to constructing a language processor.

Relationship to CS program objectives and outcomes:
This course primarily contributes to the Computer Science program outcomes that develop student:
(a) An ability to apply knowledge of computing and mathematics appropriate to the programme outcomes and to the discipline;
(c) An ability to analyse a problem, and identify and define the computing requirements appropriate to its solution.

Relationship to CS program criteria:

Criterion	DS	PF	AL	AR	OS	NC	PL	HC	GV	IS	IM	SP	SE	CN
Scale: 1 (highest) to 4 (lowest)		4	2				1						2	

Discrete Structures (DS), Programming Fundamentals (PF), Algorithms and Complexity (AL), Architecture and Organization (AR), Operating Systems (OS), Net-Centric Computing (NC), Programming Languages (PL), Human-Computer Interaction (HC), Graphics and Visual Computing (GV), Intelligent Systems (IS), Information Management (IM), Social and Professional Issues (SP), Software Engineering (SE), Computational Science (CN).

Course content distribution:

Percentage content for			
Mathematics	Science and engineering subjects	Complementary electives	**Total**
0%	100%	0%	100%

Persons who prepared this description:
Dr. Fai Wong

Part B – General Course Information and Policies

2nd Semester 2014/2015

Instructor: Dr. Fai Wong Office: E11-4010
Office hour: Mon ~ Fri 11:00 am – 13:00 pm, or by appointment Phone: 8822 4478
Email: derekfw@umac.mo

Time/Venue: Wed 11:00 – 12:45, E11-1018 (tutorial)
Fri 14:00 – 15:45, E11-1015 (lecture)

Grading distribution:

Percentage Grade	Final Grade	Percentage Grade	Final Grade
100 - 93	A	92 - 88	A–
87 - 83	B+	82 - 78	B
77 - 73	B–	72 - 68	C+
67 - 63	C	62 - 58	C–
57 - 53	D+	52 - 50	D
below 50	F		

Comment:
The objectives of the lectures are to explain and to supplement the text material. Students are responsible for the assigned material whether or not it is covered in the lecture. Students who wish to succeed in this course should read the textbook prior to the lecture and should work all homework and project assignments. You are encouraged to look at other sources (other texts, etc.) to complement the lectures and text.

Homework policy:
The completion and correction of homework is a powerful learning experience; therefore:
- There will be approximately 3 homework assignments.
- Homework is due one week after assignment unless otherwise noted, no late homework is accepted.
- The course grade will be based on the average of the HW grades.

Course project:
The project is probably the most exciting part of this course and provides students with meaningful experience to extend and enhance an existing compiler and interpreter:
- You will work with group of two students for the course project.
- The requirements will be announced and discussed in class.
- The project will be presented at the end of semester.

Exams:
One 2-hour mid-term exam will be held during the semester. Both the mid-term and final exams are closed book examinations.

Note:
- Check UMMoodle (https://ummoodle.umac.mo/) for announcement, homework and lectures. Report any mistake on your grades within one week after posting.
- No make-up exam is given except for CLEAR medical proof.
- Cheating is absolutely prohibited by the university.

Student Disabilities Support Service:
The University of Macau is committed to providing an equal opportunity in education to persons with disabilities. If you are a student with a physical, visual, hearing, speech, learning or psychological impairment(s) which substantially limit your learning and/or activities of daily living, you are encouraged to communicate with your instructors about your impairment(s) and the accommodations you need in your studies. You are also encouraged to contact the Student Disability Support Service of the Student Counselling and Development Section (SCD), which provides appropriate resources and accommodations to allow each student with a disability to have an equal opportunity in education, university life activities and services at the University of Macau. To learn more about the service, please contact SCD at scd.disability@umac.mo, or 8397 4901 or visit the following website:
http://www.umac.mo/sao/scd/sds/aboutus/en/scd_mission.php

Appendix:

Rubric for Program Outcomes

(a) An ability to apply knowledge of computing and mathematics appropriate to the programme outcomes and to the discipline

Measurement Dimension	Excellent (80-100%)	Average (60-79%)	Poor (<60%)
1. An ability to apply knowledge of computing to the solution of complex computing problems.	Students understand the computing principles, and their limitations in the respective applications. Use the computing principles to formulate and solve complex computing problems.	Students understand the computing principles, and their limitations in the respective applications. But they have trouble in applying these computing principles to formulate and solve complex computing problems.	Students do not understand the computing principles, and their limitations in the respective applications. Do not know how to apply the appropriate computing principles to formulate and solve complex computing problems.
2. An ability to apply knowledge of mathematics to the solution of complex computing problems.	Students understand the mathematical principles, e.g., calculus, linear algebra, probability and statistics, relevant to computer science, and their limitations in the respective applications. Use mathematical principles to formulate and solve complex computing problems.	Students understand the theoretical background and know how to choose mathematical principles relevant to computer science. But they have trouble in applying these mathematical principles to formulate and solve complex computing problems.	Students do not understand the mathematical principles and do not know how to formulate and solve complex computing problems.

(c) An ability to analyse a problem, and identify and define the computing requirements appropriate to its solution

Measurement Dimension	Excellent (80-100%)	Average (60-79%)	Poor (<60%)
1. An ability to understand problem and identify the fundamental formulation	Students understand problem correctly and can identify the fundamental formulation	Student understand problem correctly, but have trouble in identifying the fundamental formulation	Students cannot understand problem correctly, and they do not know how to identify the fundamental formulation
2. An ability to choose and properly apply the correct techniques	Students know how to choose and properly apply the correct techniques to solve problem.	Students can choose correct techniques but have trouble in applying these techniques to solve problem.	Students have trouble in choosing the correct techniques to solve problem.