CSCI E-95, Spring 2018: Compiler Design and Implementation

Prof. James L. Frankel
Harvard University
Pre-Class 1/23/2018

- Photos of class members, course staff, AV staff
First Class Meeting on 1/23/2018

• Questions?

• Syllabus
  • Including books

• Order books
  • Compilers: Principles, Techniques, and Tools, 2nd Ed. by Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman

• Students outside New England who will not be coming to class for the midterm exam should make arrangements now for a proctor
  • Midterm will be three hours long
  • Must start the exam from 7:40 PM ET on Tuesday, March 20, 2018 to 7:40 PM ET on Wednesday, March 21, 2018
Class Meetings

• Class is elongated
  • 7:40-10:15 PM ET in 53 Church Street, Room L01

• If interested, students are welcome to gather with us for dinner after each class
  • Dinner is an opportunity for students to socialize in an informal setting outside of class

• Possible class activities
  • Ski trip, sailing outings
Section

• Section meets immediately before class
  • 6:35-7:35 PM ET in our classroom: 53 Church Street, Room L01
  • Very important
    • Discusses concepts & issues that are not covered in class
    • Great forum for a more interactive dialog
  • Is live streamed and also recorded
HarvardKey, Using nice, Say Hello!, Student Locations, g.harvard.edu e-mail address

• Ensure that your **Harvard Key** is established
  • Login to nice.harvard.edu using SFTP/SSH (SecureCRT)

• Ensure that you have an account on the **nice** computers

• **Submit a video** using **Say Hello!** in Canvas

• Enter your location using **Student Locations** in Canvas

• Ensure that you have a g.harvard.edu e-mail address
Enroll in Piazza

• Enroll in Piazza
  • Important questions are answered in that forum

• Ask questions in Piazza so the whole class can benefit from the answers
  • Personal questions should be sent to the course staff via e-mail
  • If appropriate, include all three course staff members in e-mail to allow the fastest reply
Problem Set 0

• Go over Problem Set 0

• Complete Problem Set 0
  • Install git as described on the section web site
  • Modify the course questionnaire with your personal answers
  • Fix warnings and errors in fix-this-program.c on the nice computers
  • Write the word-count.c program
  • Create a branch named “ps0-submit”, create a pull request, add the appropriate comment, accept the pull request

• Problem Set 0 is due at midnight on Sunday night, January 28th, 2018
Problem Set 1

• Go over **Problem Set 1**
  • Due at midnight on Sunday, February 4\(^{th}\), 2018
New Material for this Week

• Cover new slides
  • Overview of Compiler
  • Review of the C Programming Language
Pre-Class 1/30/2018

- Photos of new class members
Second Class Meeting on 1/30/2018

• Questions?
  • Problem Set 0
  • Problem Set 1
  • Section or lecture

• Students outside New England who will not be coming to class for the midterm exam should make arrangements now for a proctor
  • Midterm will be three hours long
  • Must start the exam from 7:40 PM ET on Tuesday, March 20, 2018 to 7:40 PM ET on Wednesday, March 21, 2018
Modification to PS1

• Keep code in src directory
• Accept the same command line interface as does the skeleton code in the class repository
• You are not required to keep the same structure that is used by the skeleton compiler, but it is probably the correct approach unless there is a good reason to deviate from its structure

• Note that the class web site has a bullet item about the change to PS1
Problem Set 2

• Present **Problem Set 2**
  • Due at midnight on Sunday, February 18\(^{th}\), 2018

• Show the grammar for our subset of the C Programming Language on the class web site

• Problem Set 2 is perhaps the most time-consuming problem set in the course
  • Start early
  • Ask questions

• We’ll present information on yacc/bison next week
New Material for this Week (1 of 3)

• Cover new slides
  • Lexical Analysis

• Example
  • Construct an NFA from a regular expression using the McNaughton-Yamada-Thompson algorithm exactly
  • Construct a DFA from an NFA using the Subset Construction
New Material for this Week (2 of 3)

- Cover new slides
  - Using Lex
- Example
  - Show `lexer-standalone.lex & lexer.c`
  - Run `make standalone`
  - Demonstrate `./lexer`
New Material for this Week (3 of 3)

• Cover more new slides
  • Review of the C Programming Language
    • Continue from the Prefix, Infix, and Postfix Operators slide
Pre-Class 2/6/2018

• Photos of new class members
Third Class Meeting on 2/6/2018

• Questions?
  • Problem Set 1
  • Problem Set 2
  • Section or lecture

• Students outside New England who will not be coming to class for the midterm exam should have already **made arrangements for a proctor**
  • Midterm will be three hours long
  • Must start the exam from 7:40 PM ET on Tuesday, March 20, 2018 to 7:40 PM ET on Wednesday, March 21, 2018
Skeleton Compiler in the Class Repository

• Even if you’re not basing your compiler on the skeleton compiler in the class repository, you should be examining how it is constructed, built, and tested
• Read the assigned textbook readings
• Do the practice problems in the textbook
• Even though we are not assigning textbook problem sets, you are responsible for the readings
  • Useful for the programming problem sets
  • Necessary for the midterm exam
Lectures & Section

- Watch both the lecture and section videos
- All lecture and section videos are recorded
- Many questions are answered in lecture and section
Problem Set 1

• Present **Problem Set 1**
  • Was due at midnight on Sunday, February 4\(^{th}\), 2018

• Everyone should have already completed PS1
Problem Set 2 (1 of 2)

• **Problem Set 2**
  • Due at midnight on Sunday, February 18\textsuperscript{th}, 2018

• Problem Set 2 is perhaps the most time-consuming problem set in the course
  • Start early
  • Ask questions
Problem Set 2 (2 of 2)

• Every node in the AST should have a node type
  • Which operator/statement?
  • Which/how many children?
  • etc.

• Don't create unnecessary nodes
  • No node for parenthesized expression
  • No node for a choice of non-terminals (such as, a or b or c)
  • No node for terminals (in most cases)
  • etc.

• Create nodes to store identifiers as strings in the AST
• Create nodes to store literals as values in the AST
  • Integer literals as binary integral values with type
  • String literals as strings
New Material for this Week (1 of 4)

• Cover new slides
  • Syntax Analysis

• Example
  • Show recursiveDescentParser.c
  • Run make recursive
  • Demonstrate ./recursiveDescentParser
New Material for this Week (2 of 4)

• Cover new slides
  • Using Yacc

• Example
  • Show lexer.lex & parser.y
  • Run make parser
  • Demonstrate ./parser
New Material for this Week (3 of 4)

• Cover new slides
  • Parse Tree, AST, and Type Tree
  • Just look at the Parse Tree and AST portions of these slides (i.e., ignore the Type Tree slide for now)
New Material for this Week (4 of 4)

• Cover more new slides
  • Review of the C Programming Language
    • Continue from the Unary Prefix Operators slide
Pre-Class 2/13/2018

• Photos of new class members
Fourth Class Meeting on 2/13/2018

• Starting with today’s class and next week’s section, we’re switching to Zoom chat
• Student’s are also welcome to ask questions directly using their audio/video link in Zoom in the Room

• Questions?
  • Problem Set 2
  • Section or lecture
Problem Set 2

• **Problem Set 2**
  • Due this coming Sunday, February 18\(^{th}\), 2018 at midnight

• Problem Set 3 is based on the AST from PS2
Problem Set 3

• Present **Problem Set 3**
  • Due Sunday, March 4\textsuperscript{th}, 2018 at midnight

• Problem Set 3 involves building symbol tables & type data structures
New Material for this Week (1 of 4)

- Cover new slide
  - **Symbol Table Management**
  - Go over symbolTables.c
New Material for this Week (2 of 4)

• Cover new slides
  • Parse Tree, AST, and Type Tree
  • Let’s examine the Type Tree slide
  • See how the Type Tree is derived from the AST for a decl
New Material for this Week (3 of 4)

• Cover new slides
  • Type Checking
  • Cover through the Function Types slide
New Material for this Week (4 of 4)

• Continue with the C Programming Language slides
  • Review of the C Programming Language
    • Continue from after the Scope (§4.2.1) slide
    • Covered through Position of Type Qualifiers slide
Pre-Class 2/20/2018

• Photos of new class members
Fifth Class Meeting on 2/20/2018

- Reminder: We’ve switched to Zoom chat
- Student’s are also welcome to ask questions directly using their audio/video link in Zoom in the Room

- Questions?
  - Problem Set 2
    - AST
  - Problem Set 3
    - Type Trees
    - Symbol Tables
  - Section or lecture
Problem Set 2

• **Problem Set 2**
  • Was due this past Sunday, February 18\(^{th}\), 2018

• Clarification: When emitting parentheses around syntax elements in the pretty print of the AST, *never* emit parentheses where they are *not* allowed to appear in C (*i.e.*, fully-parenthesize every subexpression, but only where parentheses are legal in C)

• How is everyone doing with PS2?
Problem Set 3

• **Problem Set 3**
  • Will be due Sunday, March 4\textsuperscript{th}, 2018 at midnight

• Problem Set 3 involves building symbol tables & type data structures
Reviewing Upcoming Dates

• PS3 Due: Sunday, March 4\textsuperscript{th}, 2018 at midnight

• Spring Break: March 11-17, 2018
  • No class on Tuesday, March 13\textsuperscript{th}, 2018

• Midterm Exam: 7:40-10:40 PM ET on Tuesday, March 20\textsuperscript{th}, 2018
New Material for this Week (1 of 2)

• Cover new slides
  • Table-Driven Top-Down Parsing
New Material for this Week (2 of 2)

• Type checking slides
  • Type Checking
  • Continue after the Function Types slide

• Detour to Numeric Encodings slides
  • Detour to Character Encodings slides

• Covered through the Usual Binary Conversions (Choose first that applies) slide
Pre-Class 2/27/2018

• Photos of new class members
Sixth Class Meeting on 2/27/2018

- Reminder: *We’ve switched to Zoom chat*
- Student’s are also welcome to ask questions directly using their *audio/video link in Zoom in the Room*

- Questions?
  - Problem Set 2
    - AST
  - Problem Set 3
    - Type Trees
    - Symbol Tables
  - Section or lecture
Harbison & Steele

• Even though no specific readings from Harbison & Steele are assigned in the syllabus, everyone should be reading and referring to H&S on a regular basis

• Familiarity with Chapters 1, 2, and 4 through 9 is required and assumed
Problem Set 3

• **Problem Set 3**
  • Will be due this coming Sunday, March 4\textsuperscript{th}, 2018 at midnight

• Problem Set 3 involves building symbol tables & type data structures
Problem Set 4

• Problem Set 4
  • Will be due Sunday, March 25th, 2018 at midnight

• Problem Set 4 involves semantic analysis/type checking

• Present Problem Set 4
Reviewing Upcoming Dates

• PS3 Due: Sunday, March 4\textsuperscript{th}, 2018 at midnight

• We have just one more class meeting (on Tuesday, March 6\textsuperscript{th}, 2018) before the Spring Break and the Midterm Exam

• Spring Break: March 11-17, 2018
  • No class on Tuesday, March 13\textsuperscript{th}, 2018

• Midterm Exam: 7:40-10:40 PM ET on Tuesday, March 20\textsuperscript{th}, 2018

• PS4 Due: Sunday, March 25\textsuperscript{th}, 2018 at midnight
New Material for this Week (1 of 4)

• Type checking slides
  • Type Checking
  • Review the Minimum Integer Precision and Range (§5.1.1, p. 125 & Table 5-2, p. 127) slide
  • Continue after the Usual Binary Conversions (Choose first that applies) slide

• After the Examples of Conversions slide, discuss what happens with:

```c
int i = -5;
unsigned int ui = 6;

if(i < ui)
    printf("i < ui\n");
else
    printf("i >= ui\n");
```
New Material for this Week (2 of 4)

• Continue with the C Programming Language slides
  • Review of the C Programming Language
    • Review the Position of Type Qualifiers slide
    • Continue from after the Position of Type Qualifiers slide
New Material for this Week (3 of 4)

• Show symbol tables for the code in symbolTables.c
New Material for this Week (4 of 4)

• Look at some system header files on nice.fas.harvard.edu
  • They reside in directory /usr/include
    • /usr/include/stdint.h
    • /usr/include/limits.h
Pre-Class 3/6/2018

• Photos of new class members
Seventh Class Meeting on 3/6/2018

• We’re encouraging distance students to **share their video feed** and to **ask questions verbally** using their **audio/video link in Zoom in the Room**

• Questions?
  • Problem Set 2
    • AST
  • Problem Set 3
    • Type Trees
    • Symbol Tables
  • Problem Set 4
    • Type Checking
  • Section or lecture
Problem Set 3

• Problem Set 3
  • Was due this past Sunday, March 4\textsuperscript{th}, 2018 at midnight

• Problem Set 3 involves building symbol tables & type data structures
Problem Set 4

• Problem Set 4
  • Will be due Sunday, March 25th, 2018 at midnight

• Problem Set 4 involves semantic analysis/type checking
Midterm Exam

- Exam is **three hours in duration**
  - The exam is designed to take two hours
  - But, the exam is time-consuming – don’t worry if you take all three hours
  - The problems are scored in points where a point is weighted to be approximately one minute of exam answer duration

- **7:40-10:40 PM ET on Tuesday, March 20\(^{th}\), 2018 – two weeks from today**

- The **in-person** exam will be held in our usual classroom: 53 Church Street, Room L01

- Distance students must start the exam in the 24 hour period from 7:40 PM ET on 20-Mar-2018 to 7:40 PM ET on 21-Mar-2018

- Open book for Harbison&Steele and for Aho,Lam,Sethi&Ullman
  - **Bring these two books with you to the exam**
  - Nothing else can be used during the exam: no notes, no electronics, no computers, no cell phones, no communication
  - Nothing else is needed!
Material on Midterm Exam

• Book readings
  • Aho, Lam, Sethi & Ullman, Chapters 1-6
  • Harbison & Steele
    • Familiarity with Chapters 1, 2, and 4 through 9 is required and assumed

• Material covered in class through – and including – today’s class meeting except for Intermediate Representation and MIPS Instruction Set

• Material covered in Problem Sets 1-3

• Discussions on Piazza

• Material covered in section is beneficial, but not required
Topics for Midterm Exam (1 of 3)

- C Programming Language
- Lexing and Lex/Flex
- Regex’s
- Syntax Analysis
- Context-Free Grammars
- Leftmost and Rightmost Derivations
- Ambiguity
- Left Recursion
- Left Factoring
- Parsing and YACC/Bison
- The Grammar for Our Subset-C Language
- NFA, DFA
  - Creating an NFA from a Regex using the McNaughton-Yamada-Thompson Algorithm
  - Conversion of NFA to DFA using Subset Construction
Topics for Midterm Exam (2 of 3)

• Top-Down Parsing
• Recursive Descent Parsing
• Symbol Table Management
• Types in the C Programming Language
• Representing Types in Our Compiler
• Parse Tree, AST
• Name Spaces
• FIRST, FOLLOW, LL(1) Grammars, Predictive Parsing Table M, Table-Driven Predictive Parsing
• Number Representations
Topics for Midterm Exam (3 of 3)

• Type Checking/Semantic Analysis
• C Standard Conversions
• Bottom-Up Parsing
• Shift-Reduce Parsing
Reviewing Upcoming Dates

• This is the last class meeting before Spring Break and the Midterm Exam

• Spring Break: March 11-17, 2018
  • No class or section next week on Tuesday, March 13th, 2018

• Midterm Exam: 7:40-10:40 PM ET on Tuesday, March 20th, 2018

• PS4 Due: Sunday, March 25th, 2018 at midnight
New Material for this Week (1 of 2)

• Present the **Shift-Reduce Parsing** slides
  • Bottom-Up LR(k) Parsing

• **Shift-Reduce Parsing**

• Sufficient for all modern computer languages, including C
• Our students don’t need to be able to create an LR-parsing table – just be able to execute a parser using it
New Material for this Week (2 of 2)

• Present the **Intermediate Representation** slides
  • Covered through the **String Constants** slide
Pre-Class 3/27/2018

• Photos of new class members
Eighth Class Meeting on 3/27/2018

• Ninth class meeting if you count the midterm exam meeting

• We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room

• Questions?
  • Midterm exam
  • Problem Set 3
    • Type Trees
    • Symbol Tables
  • Problem Set 4
    • Type Checking
  • Section or lecture
Problem Set 4 (1 of 2)

• Problem Set 4
  • Was due this past Sunday, March 25\(^{th}\), 2018 at midnight

• Problem Set 4 involves semantic analysis/type checking
Problem Set 4 (2 of 2)

• Insertion of casts in PS4
  • Explicit casts are inserted into the AST in PS4 when the type of the result of an operator is not the same as the type when that result is used as an operand of another operator
  • It is possible that more than one cast would be inserted into the AST for a single implicit type conversion
    • For example, one cast might be added as a result of applying the Usual Unary Conversions and another added as a result of applying the Usual Binary Conversions
    • If this is the case, then multiple casts should be added to the AST
Problem Set 5 (1 of 2)

• Problem Set 5
  • Will be due on Sunday, April 8th, 2018 at midnight

• Present Problem Set 5

• Problem Set 5 involves generating intermediate code (IR)
Problem Set 5 (2 of 2)

• The most important design methodology is to remember to follow the approach where expressions are evaluated to an *lvalue* (when possible) and are evaluated to an *rvalue* only when necessary
  • Expressions are evaluated to an *rvalue* when:
    • The result of an operator is defined to never be an *lvalue*
    • The operand of an operator requires its operand to be an *rvalue*

• Tag each AST node in an expression with
  • The name of the temporary that holds the *lvalue* or *rvalue* of that subexpression
  • Whether that temporary is an *lvalue* or an *rvalue*
Midterm Exam

• Review the statistics from the midterm exam

• Go over the answers to the midterm exam problems
Errata from Harbison & Steele

• Link to Harbison’s errata is on the class web site

• There is also a link to our own errata from H&S
Students Not Yet Working on PS4

• Students who are not yet working on PS4 may choose the following path
  • If you have not yet started work on PS4, it is possible to delay work on PS4 or, perhaps, to never work on PS4
  • If you are certain that a type-correct program is given to your compiler, then PS4 would not change the AST in any way
  • Therefore, in these cases, we suggest that you skip working on PS4 for now and implement PS5 *before* PS4
  • This gives you a shortcut to being able to run code

• If you haven’t already done so, you will have to label every node in an expression in the AST with the type of that node
  • This would normally happen in PS4, but if PS4 is being skipped for now, it is still needed for PS5
  • This is required to know which IR instruction should be generated
New Material for this Week

• Present the **Intermediate Representation** slides
  • Start with the **IR Code for String Literal** slide
  • Covered through the **Accessing a Called Function’s Return Value** slide
Pre-Class 4/3/2018

• Photos of new class members
Ninth Class Meeting on 4/3/2018

- Tenth class meeting if you count the midterm exam meeting

- We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room

- Questions?
  - Midterm exam
  - Problem Set 3
    - Type Trees
    - Symbol Tables
  - Problem Set 4
    - Type Checking
  - Problem Set 5
    - Intermediate Representation (IR) Generation
  - Section or lecture
Problem Set 5

• Problem Set 5
  • Will be due on this coming Sunday, April 8\textsuperscript{th}, 2018 at midnight

• Problem Set 5 involves generating intermediate code (IR)
Problem Set 6

• Problem Set 6
  • Will be due on Sunday, April 22\textsuperscript{th}, 2018 at midnight

• Present Problem Set 6

• Problem Set 6 involves generating MIPS assembly code suitable for SPIM
Students Not Yet Working on PS4

• Students who are not yet working on PS4 may choose the following path
  • If you have not yet started work on PS4, it is possible to delay work on PS4 or, perhaps, to never work on PS4
  • If you are certain that a type-correct program is given to your compiler, then PS4 would not change the AST in any way
  • Therefore, in these cases, we suggest that you skip working on PS4 for now and implement PS5 before PS4
  • This gives you a shortcut to being able to run code

• If you haven’t already done so, you will have to label every node in an expression in the AST with the type of that node
  • This would normally happen in PS4, but if PS4 is being skipped for now, it is still needed for PS5
  • This is required to know which IR instruction should be generated
Clarification About Using the \texttt{phi} IR

• If you choose to use the \texttt{phi} IR rather than assign to temporaries more than once, ...

• Then, the \texttt{phi} IR would need to be used for \&\&, ||, and ? : 

• My \texttt{Intermediate Representation} slides presented last week made it seem that \texttt{phi} would be used in just the ? : operator
New Material for this Week (1 of 4)

• Present the Intermediate Representation slides
  • Review all slides involving function calling and function bodies
  • Start with the Calling Functions with No Parameters and No Return Value slide
New Material for this Week (2 of 4)

• Present the **MIPS Instruction Set** slides
New Material for this Week (3 of 4)

- Present the **MIPS Assembly Language** slides
New Material for this Week (4 of 4)

• Present the Run-time Environment slides
  • Covered through the Our Stack Frame Implementation slide
Pre-Class 4/10/2018

• Photos of new class members
Tenth Class Meeting on 4/10/2018

• Eleventh class meeting if you count the midterm exam meeting

• We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room.
Questions

- MIPS Instruction Set and Assembly Language
- Run-time Environment
- Midterm exam
- Problem Set 3
  - Type Trees
  - Symbol Tables
- Problem Set 4
  - Type Checking
- Problem Set 5
  - Intermediate Representation (IR) Generation
- Problem Set 6
  - Generating MIPS assembly code suitable for SPIM
- Section or lecture
Problem Set 5

- Problem Set 5
  - Was due this past Sunday, April 8th, 2018 at midnight

- Problem Set 5 involves generating intermediate code (IR)
Problem Set 6

• Problem Set 6
  • Will be due on Sunday, April 22\textsuperscript{th}, 2018 at midnight

• Problem Set 6 involves generating MIPS assembly code suitable for SPIM
Double-Word Alignment

• Revision of Stack Frame Format slide in Run-time Environment slides

• Now shows the possible word for double-word alignment of the stack frame
Generate Code to Deal with Integral Value Added to a Pointer

• Generate IR code to appropriately scale an integral expression that is added to a pointer
  • See Intermediate Representation slides, IR Code for Subscript Operator with Arrays slide
Accessing Static Data and Labels

• _Global_ prefix
• _UserLabel_functionName_userLabel_
• _GeneratedLabel_integer_
• _StringLabel_integer_
New Material for this Week (1 of 4)

• Continue to present the Run-time Environment slides
  • Start with the Dynamic vs. Static Old Frame Pointer Links slide
New Material for this Week (2 of 4)

• Present the **Determining Stack Offsets** slides
New Material for this Week (3 of 4)

• Present the MIPS assembly code examples and run them under SPIM
  • printint.s
  • printstring.s
  • readstring.s
  • count.s
  • count2.s
  • count3.s
  • squares.s
  • storedints.s
New Material for this Week (4 of 4)

• Present the factorial code examples
  • factorial.c
  • factorial.ir
  • factorial - Simple Code.s
    • Started covering subroutine entry code
    • Zoom failed as we started to cover the subroutine entry code
Pre-Class 4/17/2018

• Photos of new class members
Eleventh Class Meeting on 4/17/2018

• Twelfth class meeting if you count the midterm exam meeting

• We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room
Questions

• MIPS Instruction Set and Assembly Language
• Run-time Environment
  • Stack frame format and contents
  • Determining stack frame offsets
• Problem Sets
• Problem Set 6
  • Generating MIPS assembly code suitable for SPIM
• Section or lecture
  • factorial.c
  • factorial.ir
Problem Set 6

• Problem Set 6
  • Due this coming Sunday, April 22\textsuperscript{th}, 2018 at midnight

• Problem Set 6 involves generating MIPS assembly code suitable for SPIM
Final Project (1 of 2)

• Present the Final Project description

• Final Project
  • Pre-recorded video presentation to class on Tuesday, May 8\textsuperscript{th}, 2018 beginning at 6:35 PM ET
    • Class meeting will begin at 6:35 PM in our usual classroom
    • Slots for each presentation
    • Each presentation is followed by a live Q & A session
  • By 4 PM on Tuesday, May 8\textsuperscript{th}, 2018, we need to have received:
    • From all students: URL of ten-minute video presentation
    • From distance students: Google Hangouts or Skype ID for Q & A web conferencing
  • Final code is due by 2 PM on Friday, May 11\textsuperscript{th}, 2018
Final Project (2 of 2)

- The Final Project involves code optimization
  - All optimizations must be controlled by a command line switch
    - -O0 means perform no optimization
    - -O1 means perform minimal optimization
    - etc.
  - Almost all optimization will be performed at the IR level
  - Some other optimizations
    - Resetting register usage at the beginning of each statement
    - Generating more efficient code to save and restore $s and $t registers
      - That is, not just saving all $s and $t registers
    - Better register allocation (graph coloring?)
New Material for this Week (1 of 2)

• Continue to present the factorial code examples
  • factorial - Simple Code.s
  • factorial.s
New Material for this Week (2 of 2)

• Next-Use, Liveness & Register Allocation
Pre-Class 4/24/2018

• Photos of new class members
Twelfth Class Meeting on 4/24/2018

• Thirteenth class meeting if you count the midterm exam meeting

• We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room
Remaining Class Meetings

• We will be holding our last class meeting in which material will be presented, next week on Tuesday, May 1\textsuperscript{st}, 2018

• We will meet on Tuesday, May 8\textsuperscript{th}, 2018 for Final Project presentations
Questions

• MIPS Instruction Set and Assembly Language
• Run-time Environment
  • Stack frame format and contents
  • Determining stack frame offsets
• Next-Use, Liveness, Graph Coloring & Register Allocation
• Final Project
• Problem Sets
• Problem Set 6
  • Generating MIPS assembly code suitable for SPIM
• Section or lecture
  • factorial.c
  • factorial.ir
Chordal Graphs (1 of 3)

• See paper: *Register Allocation via Coloring of Chordal Graphs* by Fernando Magno Quintão Pereira and Jens Palsberg, UCLA Computer Science Department

• A graph is *chordal* if every cycle with four or more edges has a chord, that is, an edge which is not part of the cycle but which connects two vertices on the cycle. (Chordal graphs are also known as *triangulated*, *rigid-circuit*, *monotone transitive*, and *perfect elimination* graphs.)
Chordal Graphs (2 of 3)

- (a) is a Chordal Graph
- (b) & (c) are Non-Chordal Graphs
Chordal Graphs (3 of 3)

- The authors observed that the interference graphs of real-life programs tend to be chordal graphs.
- The paper reports that, for example, 95% of the methods in the Java 1.5 library have chordal interference graphs when compiled with the JoeQ compiler.
- Problems such as minimum coloring, maximum clique, maximum independent set and minimum covering by cliques, which are NP-complete in general, can be solved in polynomial time for chordal graphs [Fanica Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SICOMP, 1(2):180-187, 1972].
- This paper shows that, in particular, optimal coloring of a chordal graph $G = (V, E)$ can be done in $O(|E| + |V|)$ time.
Problem Set 6

• Problem Set 6
  • Was due this past Sunday, April 22\textsuperscript{th}, 2018 at midnight

• Problem Set 6 involves generating MIPS assembly code suitable for SPIM
Final Project

• Just **two weeks until Final Project presentations** during section and class time

• **Pre-recorded video presentation** to class on Tuesday, May 8\(^{th}\), 2018 beginning at 6:35 PM ET
  • Class meeting will begin at 6:35 PM in our usual classroom
  • Slots for each presentation
  • Each presentation is followed by a **live Q & A session**

• By **4 PM ET** on Tuesday, May 8\(^{th}\), 2018, we need to have received **via e-mail**:
  • From all students: URL of **ten-minute video presentation**
  • From distance students/students not present in person: Google Hangouts or Skype ID for Q & A web conferencing

• Final code is due by **2 PM ET** on Friday, May 11\(^{th}\), 2018
Passing Command-Line Arguments to a MIPS Program Running Under SPIM

• See argcargv.s at http://sites.fas.harvard.edu/~libe295/spring2018/argcargv.s for a program that prints out argc and each argv string

• To pass arguments using command-line version of SPIM:
  • spim "" argcargv.s a b c d

• To pass arguments using QtSpim:
  • (1) First start up qtspim
  • (2) Load the .s file to be run
  • (3) Under "Simulator", click on "Run Parameters" and enter the parameters in the "Command-line arguments to pass to program" text box
  • (4) Run the program

• Note: qtspim does not do the correct parsing into separate parameters if directories include spaces!
IR Code for Accessing Multidimensional Arrays (1 of 5)

• For,

```c
int matrix[5][6];        # matrix is a 5-by-6 array of int
matrix[1][3] = 99;
```
IR Code for Accessing Multidimensional Arrays (2 of 5)

• In C, remember that a multidimensional array is stored in memory in row-major order, so the elements of matrix are stored as:

```c
matrix[0][0]
matrix[0][1]
matrix[0][2]
matrix[0][3]
matrix[0][4]
matrix[0][5]
matrix[1][0]
matrix[1][1]
matrix[1][2]
matrix[1][3]
matrix[1][4]
matrix[1][5]
matrix[2][0]
...
```
IR Code for Accessing Multidimensional Arrays (3 of 5)

• Starting with our example,

```c
int matrix[5][6];        # matrix is a 5-by-6 array of int
matrix[1][3] = 99;
```

• is syntactic sugar for,

```c
int matrix[5][6];        # matrix is a 5-by-6 array of int
*(*(matrix+1)+3) = 99;
```
• Once we apply type checking to,

```c
int matrix[5][6];  // matrix is a 5-by-6 array of int
*((*(matrix+1))+3) = 99;
```

• we have,

```c
int matrix[5][6];  // matrix is a 5-by-6 array of int
*( (* int)( *((pointer to array of 6 ints)matrix+1) )+3) = 99;
```
IR Code for Accessing Multidimensional Arrays (5 of 5)

• Generating code for,

```c
int matrix[5][6];  // matrix is a 5-by-6 array of int
*( (* int)( *((pointer to array of 6 ints)matrix+1) )+3) = 99;
```

• the result is,

```c
(addressOf, t1, matrix)  // t1 is an lvalue for matrix; t1 -> matrix[0][0]
  (constInt, t2, 1)        // t2 is an rvalue for 1; t2 <- 1
  (constInt, t3, 24)       // t3 is an rvalue for 24 (6*sizeof(int)) (i.e., sizeof(array of 6 int)); t3 <- 24
  (addSignedWord, t5, t1, t4)  // t5 is an rvalue for (pointer to array of 6 ints)matrix+1
    (constInt, t6, 3)  // t6 is an rvalue for 3; t6 <- 3
    (constInt, t7, 4)  // t7 is an rvalue for 4 (i.e., sizeof(int)); t7 <- 4
    (multSignedWord, t8, t6, t7) // t8 is an rvalue for 3*4; t8 <- 3*4
    (addSignedWord, t9, t5, t8) // t9 is an rvalue for ( * int)( *(pointer to array of 6 ints)matrix+1 )+3
      (constInt, t10, 99) // t10 is an rvalue for 99; t10 <- 99
      (storeWord, t9, t10) // *( ( * int)( *(pointer to array of 6 ints)matrix+1 )+3) <- 99
```
New Material for this Week (1 of 2)

• Optimization
  • Peephole optimizations
  • Optimizations within basic blocks
  • Global optimization
  • Interprocedural optimization
New Material for this Week (2 of 2)

• Dependencies, Instruction Scheduling, Optimization, and Parallelism
  • Introduction
  • Skipped over Pipelining and Caching
  • Covered through Complications in Determining Data Dependence slide
Pre-Class 5/1/2018

• Photos of new class members
Thirteenth Class Meeting on 5/1/2018

• Fourteenth class meeting if you count the midterm exam meeting

• We’re encouraging distance students to share their video feed and to ask questions verbally using their audio/video link in Zoom in the Room
Remaining Class Meeting

• Today is our last class meeting in which material will be presented

• We will meet next week on Tuesday, May 8\textsuperscript{th}, 2018 for Final Project presentations
Questions

• Final Project/Optimizations
• MIPS Instruction Set and Assembly Language
• Run-time Environment
  • Stack frame format and contents
  • Determining stack frame offsets
• Next-Use, Liveness, Graph Coloring & Register Allocation
• Final Project
• Problem Sets
• Problem Set 6
  • Generating MIPS assembly code suitable for SPIM
• Section or lecture
  • factorial.c
  • factorial.ir
Problem Sets

• All Problem Sets have already been due
Final Project

- Just one week until Final Project presentations during section and class time
- Pre-recorded video presentation to class next week on Tuesday, May 8\textsuperscript{th}, 2018 beginning at 6:35 PM ET
  - Class meeting will begin at 6:35 PM in our usual classroom
  - Slots for each presentation
  - Each presentation is followed by a live Q & A session
- By 4 PM ET on Tuesday, May 8\textsuperscript{th}, 2018, we need to have received via e-mail:
  - From all students: URL of ten-minute video presentation
  - From distance students/students not present in person: Google Hangouts or Skype ID for Q & A web conferencing
- Final code is due by 2 PM ET on Friday, May 11\textsuperscript{th}, 2018
Final Project Presentation Slots

• Sign up now!
Class Dinner After Final Project Presentations

• All class members (and a significant other) are welcome to join us for dinner after Final Project Presentations next week

• Dinner at the Border Café
Class Sailing Trip After the Semester Ends

• I’ll send e-mail to the class when a date(s) has been set

• Interest?
Upcoming Classes

• Fall 2018: Principles of Operating Systems
  • Concepts, issues, and implementation of an operating system
  • Culmination is OS for an ARM processor including shell, memory allocation, device drivers, device-independent I/O, file system, and multiprocessing

• Spring 2019: Computer Architecture
  • Study of how computers function
  • Course begins with principles of electricity, digital logic, data path for a computer, control logic for a computer, fluency with VHDL, instruction set design, and includes the implementation of a simple computer from digital logic on an FPGA
  • Advanced topics: Caching, Pipelining, VM, MMU, Vector CPU, VLIW, MPP
New Material for this Week (1 of 2)

• Optimization
  • Show how factorialTailRecursive.c is optimized into factorialTailRecursiveOptimized.c
New Material for this Week (2 of 2)

• Dependencies, Instruction Scheduling, Optimization, and Parallelism
  • Present Pipelining and Caching
  • Start with Complications in Determining Data Dependence slide
Pre-Class 5/8/2018

• Photos of new class members
Final Class Meeting on 5/8/2018

• Please connect to share your video feed and to ask questions verbally using their audio/video link in Zoom in the Room
Final Project

• Final code is due by 2 PM ET on Friday, May 11th, 2018
Class Dinner After Final Project Presentations

• All class members (and a significant other) are welcome to join us for dinner after Final Project Presentations next week

• Dinner at the Border Café
Class Sailing Trip After the Semester Ends

• I’ll send e-mail to the class when a date(s) has been set

• Interest?
Upcoming Classes

• Fall 2018: Principles of Operating Systems
  • Concepts, issues, and implementation of an operating system
  • Culmination is OS for an ARM processor including shell, memory allocation, device drivers, device-independent I/O, file system, and multiprocessing

• Spring 2019: Computer Architecture
  • Study of how computers function
  • Course begins with principles of electricity, digital logic, data path for a computer, control logic for a computer, fluency with VHDL, instruction set design, and includes the implementation of a simple computer from digital logic on an FPGA
  • Advanced topics: Caching, Pipelining, VM, MMU, Vector CPU, VLIW, MPP
Final Project Presentations!