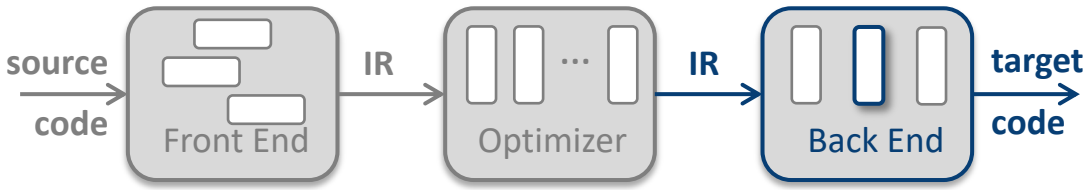
RICE

COMP 412

Local Register Allocation

Overview of the Assignment

Comp 412, Lab 2



Read excerpt from EaC3e

Dates and Deadlines

Here are the critical dates and deadlines for Lab 2

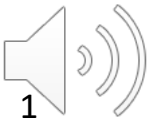
- Your final code is due at 11:59PM on Friday, October 21, 2022
  - Proof of passing code check 1 is due on Tuesday October 4
  - Proof of passing code check 2 is due on Thursday, October 13
  - The TAs will post directions for how to pass the code check
- The lab report, which consists of a questionnaire and a graph, is due at 11:59PM on Monday, October 24, 2020

Early bonus on code: 2 points per day, up to a maximum of 4 points

Late penalty on code: 2 points per day, up to a maximum of 14 points

- *No code will be accepted after Monday, October 24, 2020*

There are no bonuses or grace days for the code checks or the report.



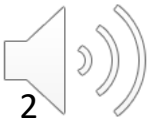
Policies

Honor Policy is the same as in Lab 1

- You may ***discuss*** anything
- Any code you that you submit for grading must be your own.
  - You should use your own code from Lab 1.
- You may collaborate with others on the makefile and shell scripts

Grading Rubric

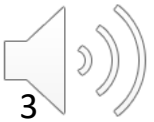
- Code Check 1 and 2 each account for 10% of the code points
- Conformance to the specs accounts to 10% of the code points
- Correctness accounts for 30% of the code points
- Effectiveness accounts for 20% of the code points
- Efficiency and scalability account for 20% of the code points



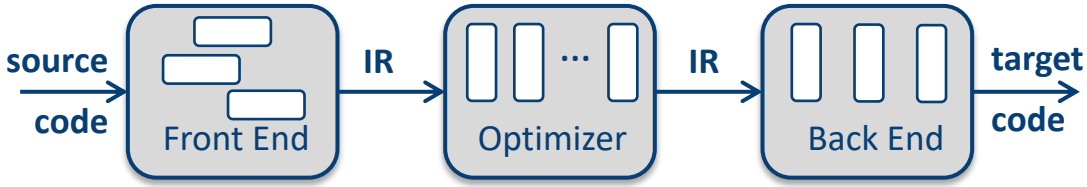
Advice

Before you start coding:

- Read the entire handout
  - Your data structures must accommodate both renaming and allocation
- Read the excerpt from Chapter 13 of EaC**3e**
  - Ignore the local allocation section in Chapter 13 of EaC**2e**
- Think about what code you can add that will simplify debugging
  - Your code will not run the first time. Design it to help with debugging.
  - For me, this issue was more important on the allocator than the renamer. Lab2_ref includes code, commented out at the moment, that verifies the correctness and completeness of the various maps at the end of each major iteration.
    → *("map" is a conceptual term; lab2_ref uses vectors and indices, not hashing.)*
- Start now. Most students find that it takes them three weeks to get to the point where they are happy with their results.

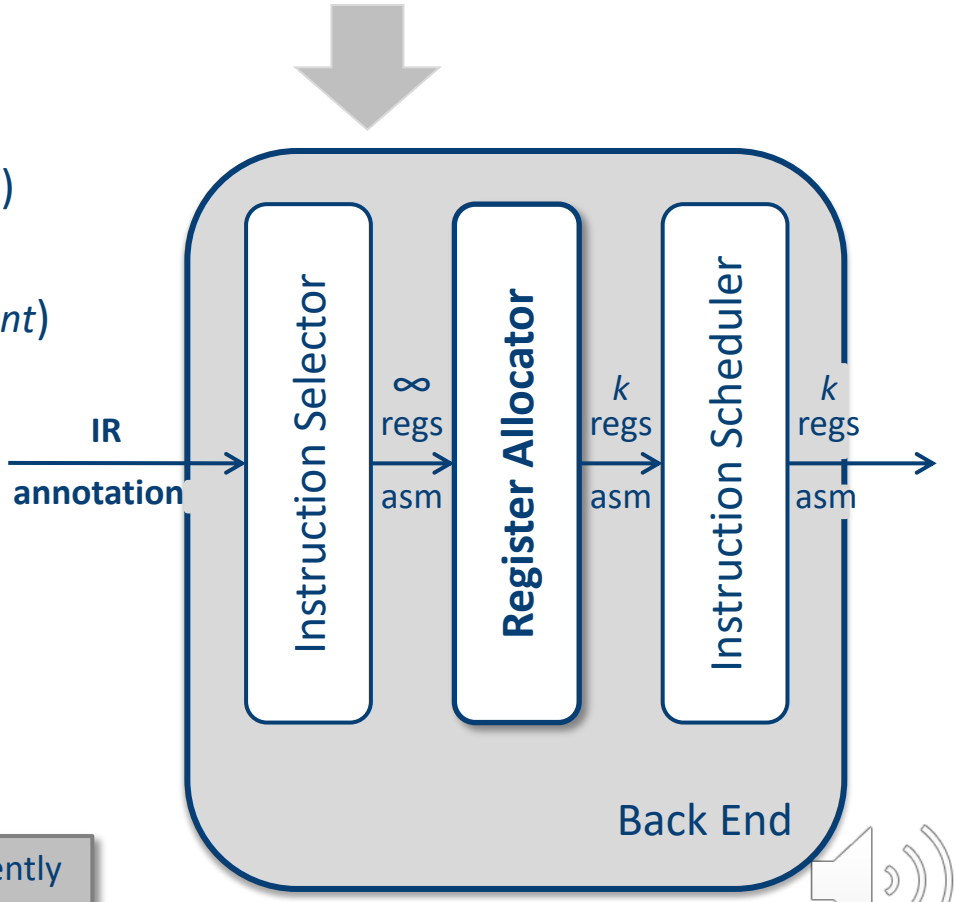


The Back End



Register Allocator:

- Decides, at each operation, which values reside in registers (*allocation*)
- Decides, for each **en**registered value, which register it occupies (*assignment*)
- Inserts, for each **un**registered value, code to spill it & restore it
- Tries to reduce spill & restore costs
- Operates efficiently
  - **O**(n), **O**(n $\log_2$ n), maybe **O**(n^2)
  - Not **O**(2^n)



Notation: The literature on register allocation consistently uses ***k*** as the number of registers on the target system.

Critical Terminology:

Virtual Registers, Physical Registers, & Live Ranges

A virtual register (VR) is a register name that the compiler uses in its internal representation of the code

- *Virtual* is used as in *virtual memory* or *virtual address*, not *virtual reality*
- Using virtual registers in the **IR**, rather than physical registers, lets the compiler defer allocation decisions & simplifies the compiler
  - *Correctly separates concerns over optimization and allocation* [*Backus*]

A physical register (PR) is a name for an actual target machine register

- *Limited supply of **PR**s, cannot add more*
- *Valid assembly code cannot name more PRs than machine supports*

A live range is a distinct value

- A live range starts with the creation of a value, or its *definition*
- A live range ends with the last use of a value

Your register allocator will find live ranges and assign them unique **VR**s

The Big Picture

Lab 2 has two major components

- A **renaming** pass
  - Find distinct *live ranges* in the block (*values*)
  - Assigns each live range a unique *virtual register* name
  - Annotates each *definition* and each *use* with the distance to the next *use*
  - Renaming is a simple linear-time pass over the code

Builds a map from LR to VR

- An **allocation** pass
  - Assigns **VR**s to **PR**s
  - Discovers points in the code where |*values*| > |*registers*|
  - At those points, chooses one or more registers to move out of registers
  - Inserts code to move values from register to memory (a *spill*) and to move them back from memory into a register (a *restore*)

Builds a map from VR to PR

- The renaming pass is tested in Code Check 1 & reused in Lab 3
  - Save a checkpoint after Code Check 1 (for use in Lab 3)
- The allocator is the harder part of the lab

Software Tools for Lab 2

A number of tools are available to help you build and debug Lab 2

- An **ILOC** Simulator
  - Essentially, an interpreter for Lab 2 ILOC
  - Matches the Lab 2 specs
    - → Single functional unit, latencies as specified, **ILOC** Lab 1 subset enforced
  - Implements **–x** command line option for code check 1
  - Documentation and a video are available on the course site.
- Lab 2 Reference Implementation
  - A functioning Lab 2 allocator, available as an executable black box
  - Has additional functionality relative to your allocator
  - Runs **only** on CLEAR
- Testing and timing scripts, and a stripped-down autograder
- A library of **ILOC** programs

All of these tools are found on **CLEAR** in ~comp412/students/lab2

What Matters?

In Lab 2, 30% of the points depend on the correctness of your allocator and 20% to the quality of the allocated code that your lab produces

- We assign these points using your allocator and the **ILOC Simulator**[1]
  - Take the unallocated code & run it
  - Take the allocated code & run it
  - Make sure the answers are identical (*correctness*)
  - Look at the number of cycles used by the allocated code (*effectiveness*)
- The goal of Lab 2 is to minimize the cost of the spill code
  - The credit for effectiveness is based on cycles used by the allocated code
  - Register allocation can introduce a **lot** of spill code
  - Attention to detail can reduce spill costs
  - **But**, you cannot always win
- Allocator speed and scalability determine 20% of the points
  - Details of the speed rubric are in the Lab Handout

Optimizations

The only optimizations allowed are register allocation and NOP elimination

- No other optimizations allowed!
- Specifically, no constant propagation, value numbering, etc. You might be tempted, as some code blocks define the input values. Don't do it!
- The code you generate must perform all the arithmetic operations that the original code does

```
loadI 412    => r0
loadI  26    => r1
add   r0,r1  => r2
loadI 0      => r0
store r2     => r0
output 0
```


```
loadI 438    => r2
loadI 0      => r0
store r2     => r0
output 0
```

This transformation would be illegal.

Pro Tips

Lab 2 & Lab 3 are open-ended, in the sense that you can often improve performance by working on the lab for a while longer

- Make managerial decisions about how to spend your time
  - Get something working by the code check
    - → *Correctness is the goal, not a great allocation*
  - Tough decision at the early code due date
- Take steps to protect against your own tinkering
  - Checkpoint your code early and often
    - → *Each time something is working, checkpoint it*
    - → *Use a version control system if you aren't already*
  - Make notes with each checkpoint so that you will know what you have
    - → *Notes and comments protect you from memory loss & exhaustion*
    - → *Your commit messages should be informative*
- Automate your testing
  - Use one of the scripts that iterates over a directory of test codes

COMP 412 Lab 2

What should you do next?

- Read the documents
  - Lab 2 handout
  - Excerpt from Chapter 13, EaC3e
- Watch the other videos
  - Lab 2 Renamer
  - Lab 2 Allocator
  - If you find **ILOC** confusing, see the video on the **ILOC** Virtual Machine
- Get started
  - Code check 1 is soon
  - Most students find that they need three weeks to get their allocator to the point where they are comfortable submitting it
- Ask questions. In office hours. In the online Q&A forum.
  - This stuff is complicated. Don't be embarrassed. Instead, be informed.