Today

- Python wrap-up
- Ch1, Ch2, Ch3 overview
- Goals for remainder of semester


Second half of Monday’s lecture

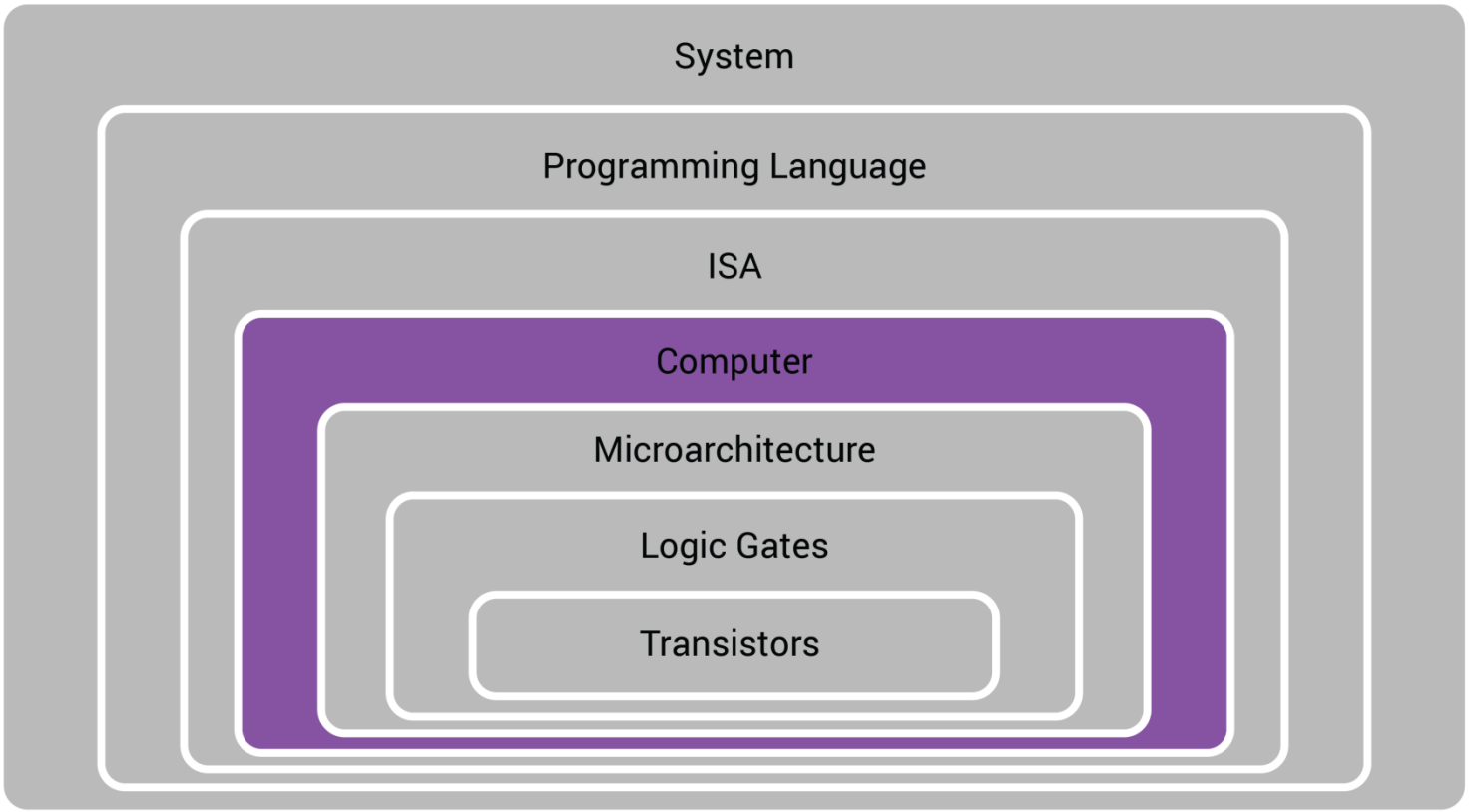
Python summary

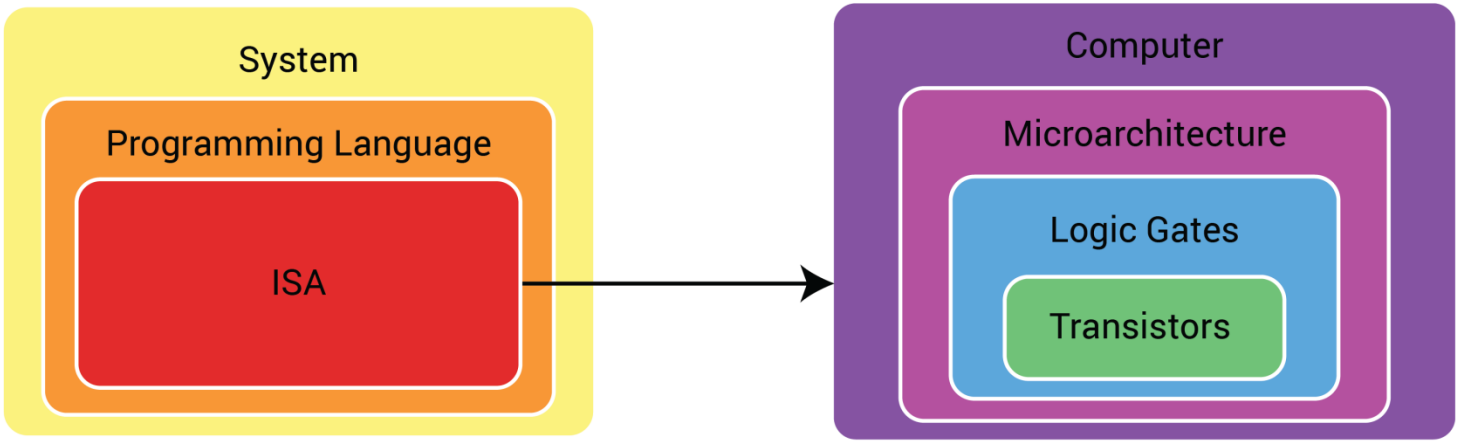
- Why Python?
  - To introduce the concept of “How to program”
  - Our goal is to understand ISA – Python is the gentle introduction to it
- Program is set of lines
  - Each line is assignment, print, control-flow, or def
  - if/else and while implement control-flow

Python FAQ

1. Can “Full” Python language can do more?
   - Yes
2. I can do <foobar> in Python, why won’t simulator allow it?
   - This is intentional
3. Why are we talking about ISA?
   - That is how the machine works

Questions about Python?


System
Programming Language
ISA
Computer
Microarchitecture
Logic Gates
Transistors


System
Programming Language
ISA
Computer
Microarchitecture
Logic Gates
Transistors

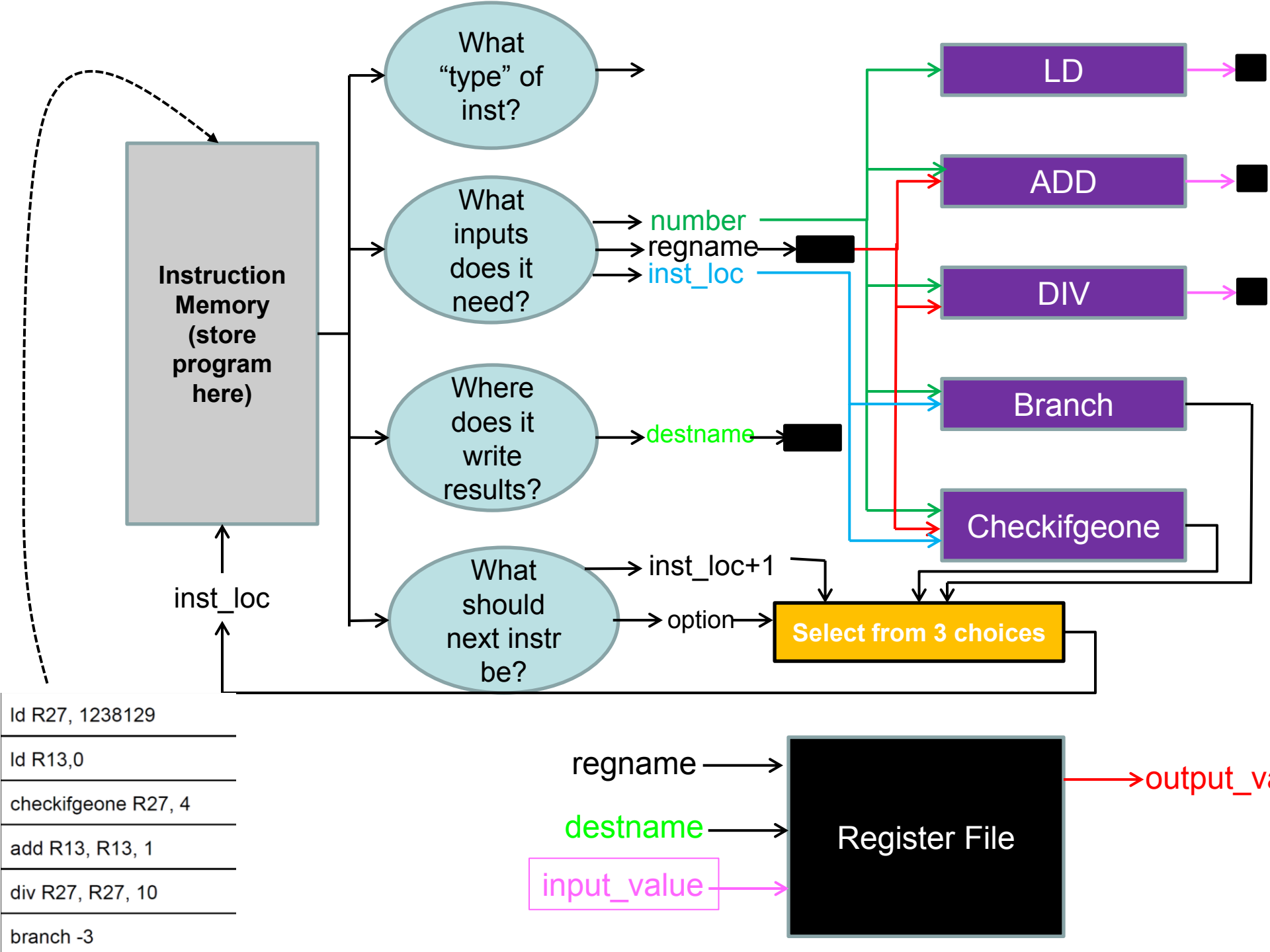
Exam 1

- Monday
- Friday: discussion-section/review
- Topics: Ch1, Ch2, Ch3
  - There will NO question asking you to write a complete Python program (I think ☺)
  - Questions?

Rest of the Semster

- Numbers – representing everything as numbers
- ISA
- Microarchitecture
- Gates

Python	ISA
n = 1238129	ld R27, 1238129
digit_count = 0	ld R13,0
while(n >= 1)	checkifgeone R27, 4
digit_count = digit_count + 1	add R13, R13, 1
n = n/10	div R27, R27, 10
	branch -3


Instruction Memory (store program here)
What "type" of inst?
What inputs does it need?
Where does it write results?
What should next instr be?
number
regname
inst_loc
destname
inst_loc+1
option
inst_loc
LD
ADD
DIV
Branch
Checkifgeone
Select from 3 choices
ld R27, 1238129
ld R13,0
checkifgeone R27, 4
add R13, R13, 1
div R27, R27, 10
branch -3
regname
destname
input_value
Register File
output_va

Friday

- Review of topics
- Work out a couple of problems
- Come prepared with questions
- Office hours will be held in class at end of lecture

September 23,

Announcements

Homework 2 due today
Homework 3 assigned today
Arrays and functions has problems

Today

Wrap up python
Chapter 1 and 2 wrap up, and chapter 3 overview

Python summary

- Why python?
  - We want to introduce “how to program”
  - Ultimately, we really want to understand ISA. Python is an easier introduction to that.
- Program is set of lines
  - Each line is assignment, print, control flow, or def.
  - if/else and while implement control-flow
    - This is the distilled aspects of a program, which is why we restrict what we do in python to this for this class.
- Python FAQ
  - Yes, Python can do more.
  - We intentionally disable these things, to push the purpose of ISA simplification.
    - This is how the machine works

Quiz 1

- Only ISA stuff from chapter 1 will be on the test
- Mainly through programming language.
- Topics: Chapter 1, chapter 2, chapter 3. Homeworks 1, 2, and 3
- No complete python programming questions

Rest of the semester

- Numbers
- ISA
- Microarchitecture
- Gates

Python to ISA

Python	ISA

n = 1238129	ld R27, 1238129
digit_count = 0	ld R13, 0
while(n >= 1)	checkifgeone R27, 4
digit_count = digit_count + 1	add R13, R13, 1
n = n/10	div R27, R27, 10
	branch -3

Sometimes, you can come out with a heuristic for assigning registers, but for now, we are going to assume they are assigned at random.

For instructions, we determine a structure. THe first column was going to be the instruction, the second column was the destination, the third was some source if needed, and the last number was an immediate field. Four numbers

Now, if we have a sequence of numbers, we can do conversion of ISA to a number.

So, what about if we want to check against four? Well, this leads to Reduced Instruction Set Computing vs Complex Instruction Set Computing. This leads to a translator to break down this bigger instruction to simpler instructions (Macro ops decoding to micro ops).

For example, if we wanted to do a complex compare against 4, we could break it down into a subtract 3 and the check against one as shown above.

Finally, returning to ISA to numbers. When we do this, it is called compiling to *binary*. Then how does the computer actually run this?

- The instructions are loaded in to Instruction Memory
- The location pointer for the current instruction (inst_loc)
    - This we will call Program Counter later in the course (PC)
- We can then read an instruction from the current inst_loc.
- We need to answer four basic questions about the instruction we retrieve
    - What type of instruction is this?
    - What inputs does it need?
        - Does it need a number? A register? The instruction location where it is?
        - It might need more than just one of these!
    - Where does it write the results?
    - What should the next instruction be?
        - It depends! If could be the next instruction location, or maybe the branch sets it to something else.

- So now we need to connect all of this up.
    - All of instructions need a number
    - Add, Div, Checkifgeone all need access to the register file
        - Register file stores values based on the different register locations.
    - Ld, Add, Div all store back to another register
        - Using destname and input_value in the register file, we can store a value there too
    - What should the next instruction be?
        - Branch and checkifgeone both produce a possible result
        - inst_loc + 1 is another possibility.
        - Based on what is executed, we need to choose between those 3

ISA to numbers, numbers to microarchitecture, microarchitecture to gates.