Mastery-Based Learning in Undergraduate Computer Architecture

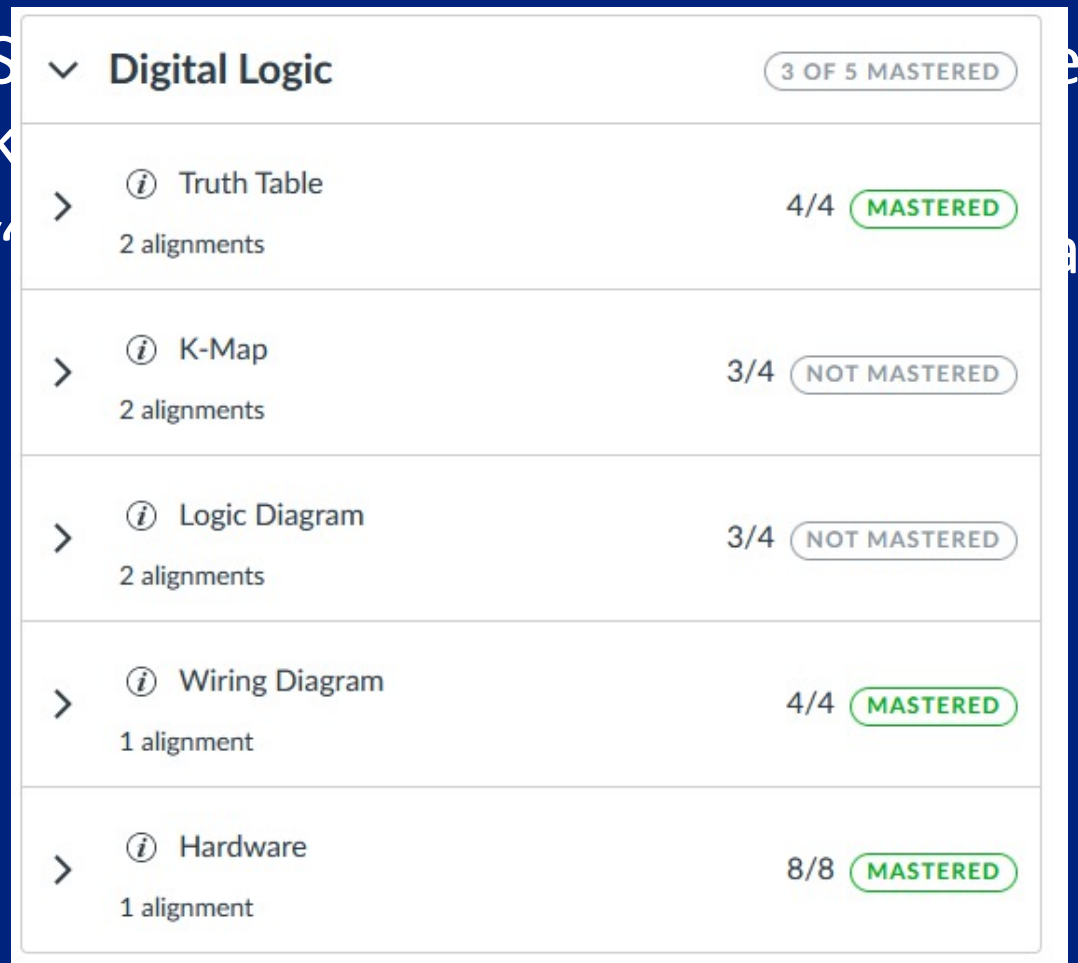
Ellen Spertus
Mills College

Zachary Kurmas
Grand Valley State University

Workshop on Computer Architecture Education 2021

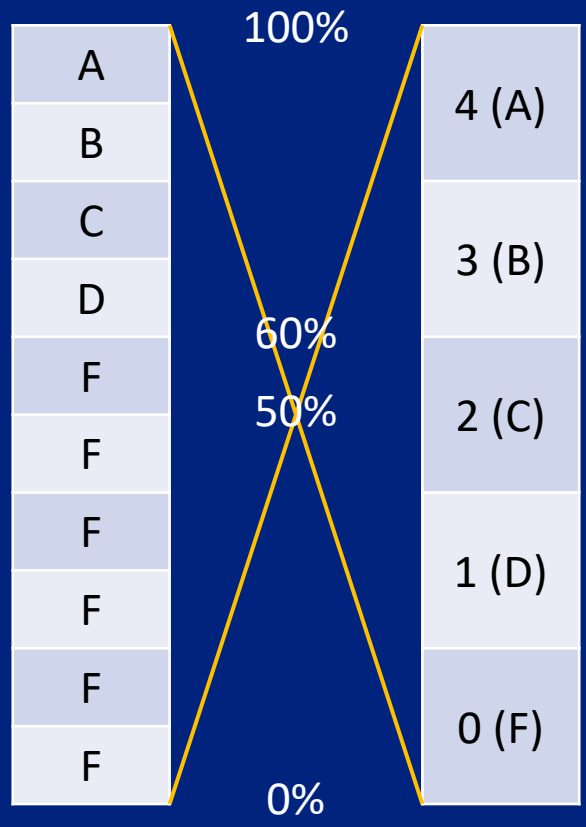
Mastery-based grading

- Focus is on learning outcomes rather than points
- S es to demonstrate their k
- “ ades



Mastery-based grading

- Focus on learning outcomes rather than points
- Allow students multiple opportunities to demonstrate their knowledge
- Don't grade students on “soft skills”
  - attendance
  - class participation
  - timeliness
- Rethink the grading scale



Example: assembly language programming

Traditional grading guide

label at start (1)

save registers (2): .5 points each

set up arguments (3): 1 point each

call other procedure (1)

restore registers (1)

restore stack pointer (1)

return to caller (1)

Mastery-based grading

Grade	Able to implement…
A	Non-leaf procedure with complex control flow
B	Non-leaf procedure without complex control flow
C	Simple leaf procedure

Challenges

- Soft skills are important
- Creating assessments
- Grading assessments

Use automation to generate, give, and grade assessments

python

Respondus®

CANVAS

GitHub

Using learning outcomes

- Provide learning outcomes for course
  - digital logic
  - computer arithmetic
  - assembly language
  - Verilog
  - processors
  - memory hierarchy
- Provide ways of demonstrating mastery of each outcome
  - lab
  - homework
  - single test
  - automatically-generated tests

Computer arithmetic learning outcomes

Computer Arithmetic
Converting between bases 2, 8, 10, and 16
Converting between decimal and two's complement
Comparing two's complement numbers
Converting between decimal and IEEE FP formats
Comparing floating-point (FP) numbers
Choosing best numeric representation given range, accuracy, and performance requirements

Computer arithmetic learning outcomes

Computer Arithmetic
Converting between bases 2, 8, 10, and 16
Converting between decimal and two's complement
Comparing two's complement numbers
Converting between decimal and IEEE FP formats
Comparing floating-point (FP) numbers
Choosing best numeric representation given range, accuracy, and performance requirements

The decimal number 122 can be represented as _____ (binary), _____ (octal) and _____ (hexadecimal).

Computer arithmetic learning outcomes

Computer Arithmetic
Converting between bases 2, 8, 10, and 16
Converting between decimal and two's complement
Comparing two's complement numbers
Converting between decimal and IEEE FP formats
Comparing floating-point (FP) numbers
Choosing best numeric representation given range, accuracy, and performance requirements

The following numbers are in two's complement notation. Please order them from lowest (most negative) to highest (most positive).

A = 00111101
B = 11100000
C = 10110101
D = 00101100

For full credit, answer the problem without converting all of the numbers to decimal.

Order:

Explain your reasoning:

Computer arithmetic learning outcomes

Computer Arithmetic
Converting between bases 2, 8, 10, and 16
Converting between decimal and two's complement
Comparing two's complement numbers
Converting between decimal and IEEE FP formats
Comparing floating-point (FP) numbers
Choosing best numeric representation given range, accuracy, and performance requirements

You are working for the United States census on a system that tells you how many people share given characteristics, such as being born in a certain year and living in California. The total US population is about 330,000,000. You never need to deal with numbers bigger than that. You are asked to evaluate the suitability of different representations for these quantities.

Which one type (`byte`, `short`, `int`, `long`, `float`, `double`, `BigInteger`, or `BigDecimal`) gets your highest recommendation?

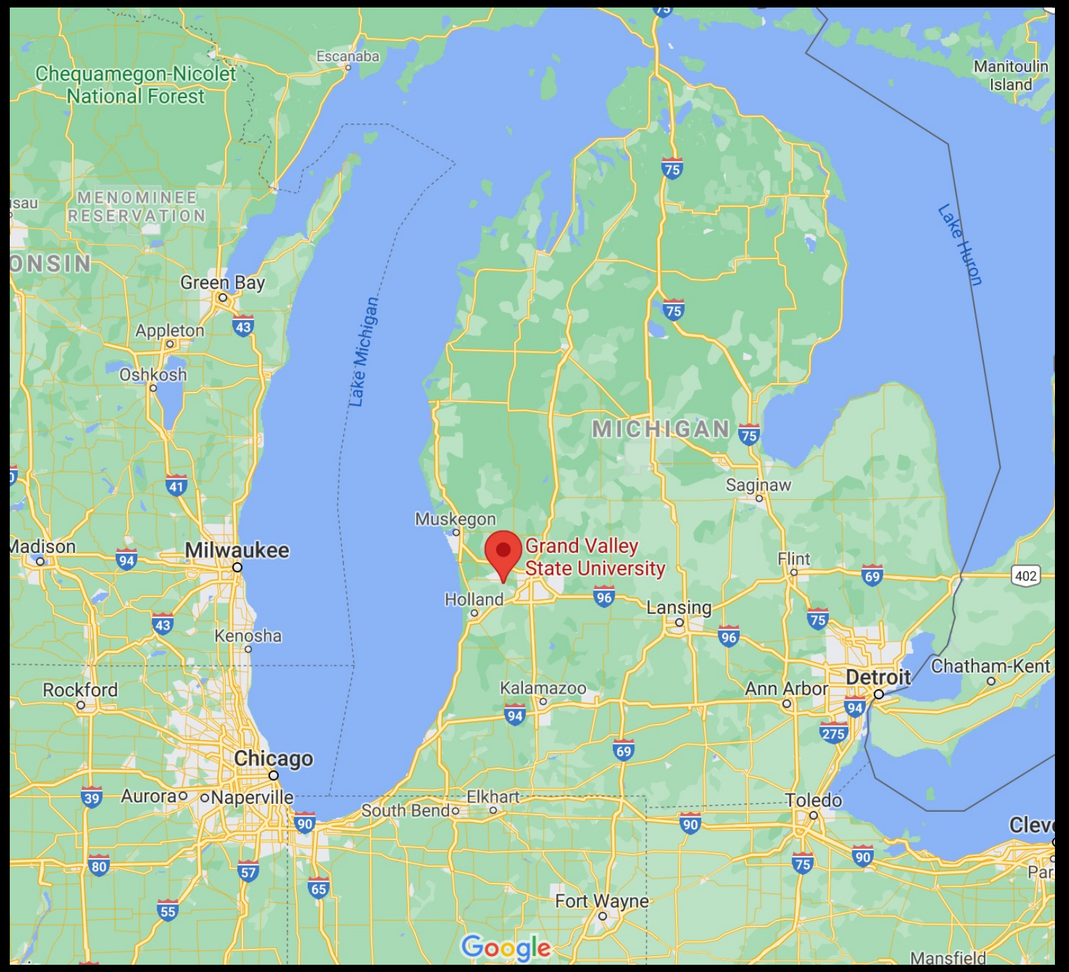
Explain your choice:

Results

Group/Outcome	4 (A)	3 (B)	Max tries
Computer Arithmetic			
Converting between bases 2, 8, 10, and 16	6	–	1
Converting between decimal and two's complement	5	1	4
Comparing two's complement numbers	6	–	3
Converting between decimal and IEEE FP formats	6	–	4
Comparing floating-point (FP) numbers	6	–	3
Choosing best numeric representation given range, accuracy, and performance requirements	4	1	3
Processors			
Encoding MIPS instructions	5	1	3
Memory Hierarchy			
Computing average memory access time	5	–	7

Grand Valley State University

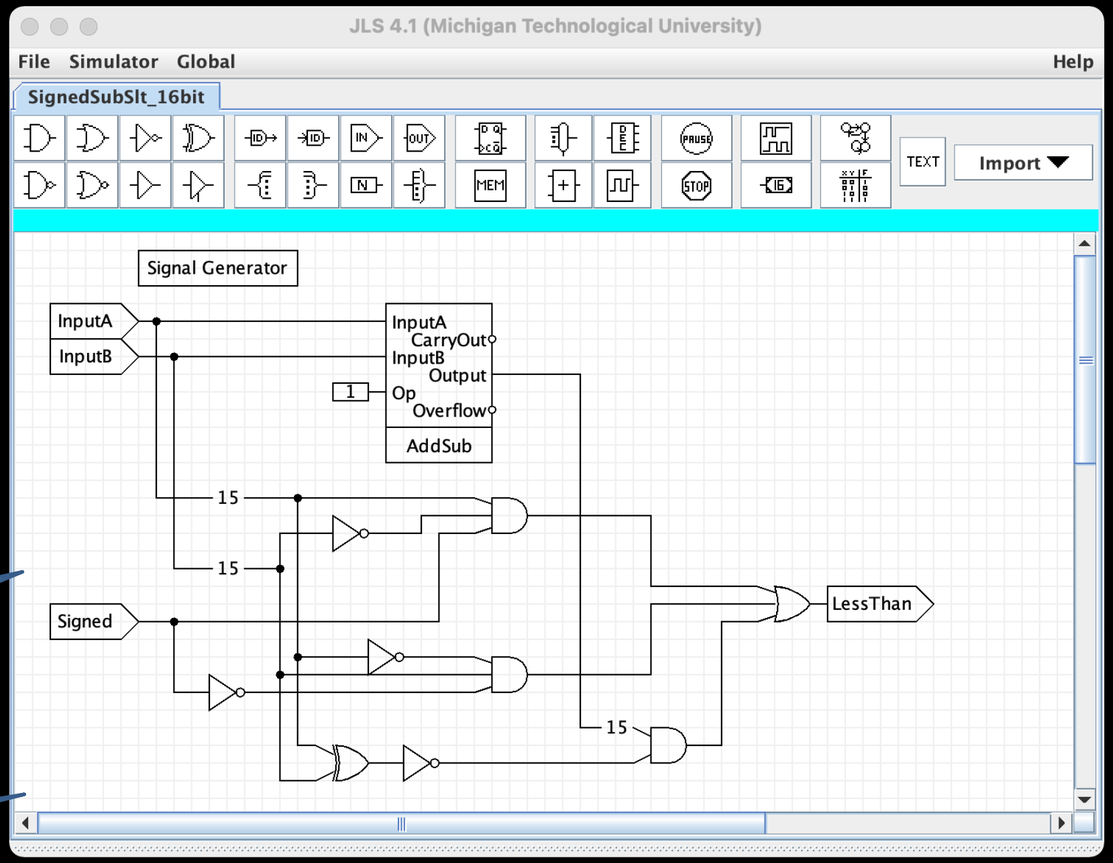
- Public, Master's comprehensive university with about 24,000 students
  - Approximately 400 Computer Science Majors
- Vast majority of CS graduates become software developers
- When doing personal CS-related projects, our students almost always choose something related to software development
  - (especially web development)



GRAND VALLEY STATE UNIVERSITY

My Computer Organization Projects

1. Build a signed adder (including overflow detection)
2. Build a
   a. Signed subtractor using the adder
   b. Signed and unsigned comparator using the subtractor
3. Build a simple ALU
4. Implement the single-cycle CPU presented in the Patterson and Hennessy text



JLS is a competitor to the more popular *Logisim*

I use JLS throughout the semester. I don't switch to an HDL.

Grand Valley State University

Initial Experience

- Submissions during first few semesters were terrible
  - Way too many bugs
  - Exams showed that they understood the material at a high level
- Fundamental problem: When partial credit offered, students would opt not to complete the more difficult/subtle features
  - Adjusting weights on features made them feel “overweight”

My Solution

- Projects don’t receive a grade until *all* my tests pass
- This is a type of mastery-based grading:
  - Students must master the project objectives before receiving a grade
- However, projects still have deadlines
  - 20% of the grade is for “timeliness”
  - Grade goes down 1-2% per day for about 14 days
  - After 14 days, maximum possible grade is 80%
- Benefits:
  - There is significant incentive for students to “master” material in a timely manner; but,
  - Students who struggle and fall behind can still earn a reasonable grade

Test-Driven Development (TDD)

- Basic workflow:
  - Write tests before writing any code
  - Write the code and debug until the tests pass
- Students must write their own tests because they need all the practice they can get
  - The only way to learn to test well is to write *a lot* of tests
- Benefits of TDD for Computer Organization projects
  - If test aren't written first, they won't get written
    - At least not in a thoughtful manner
  - Implementing with out planning ahead can make it difficult to cleanly integrate the last few features/requirements

My Project Workflow

- Students write tests for the project
- Students implement the required circuits
- Submit to the autograder when all student tests pass
- If the submission fails the “hidden”, then the students
  - Write at least one failing test
  - Fix the circuit
  - Resubmit

`DLUnit` uses `JUnit` syntax to test circuits

`MUnit` uses `JUnit` syntax to test circuits

Benefits of TDD

- The process of writing tests will
  - highlight aspects of the project that are unclear
  - Help the student create a list of tasks that must be accomplished
- There is almost never time to write tests at the end
- Tests written afterward tend to have a “confirmation bias”

Why TDD in a Computer Organization Class?

- Because they need all the testing practice they can get
- You don’t learn to be a good tester by sitting through a two-week unit in a software engineering class.
  - You have to actually write tests.
  - A lot of tests

WCAE 2021

Mastery-Based Learning in Undergraduate Computer Architecture

Ellen Spertus Mills College

Zachary Kurmas Grand Valley State University