CSE 390B, Winter 2022

Building Academic Success Through Bottom-Up Computing

Midterm Mock Exam, Test-Taking Strategies

Mock Midterm, Test-Taking Strategies Reflection, Mock Exam Walkthrough

Mock Exam

- This exam is closed-note, closed-book
  - You may only use the midterm reference sheet available under the “Resources” page on our course website
- Questions are not necessarily in order of difficulty
- You have 25 minutes to complete the exam
  - We will give you a 5-minute warning
- BREATHE.

Evaluating your test-taking strategies…

Mock Exam Debrief & Reflection

- What did you learn about yourself through this process? About your test-taking practices?
- What are two test-taking strategies that you would like to engage with in your next exam? Why?
- What is one thing that can help you relax and calm down before or during your exam?

Question 1: Circuit Design

Part a: Truth Table

A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

11 -> 10 -> 01 -> 00 -> 11

Question 1: Circuit Design

Part a: Truth Table

A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

11 -> 10 -> 01 -> 00 -> 11

Question 1: Circuit Design

Part a: Truth Table

A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

$A_{t+1} = (A_t \& B_t)$

$A_{t+1} = (\sim A_t \& \sim B_t)$

Part b: Boolean Expressions

Question 1: Circuit Design

Part a: Truth Table

A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

$A_{t+1} = (A_t \,\&\, B_t)$

$A_{t+1} = (\sim A_t \,\&\, \sim B_t)$

Part b: Boolean Expressions

$A_{t+1} = (A_t \,\&\, B_t) \mid (\sim A_t \,\&\, \sim B_t)$

Question 1: Circuit Design

Part a: Truth Table

A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

$A_{t+1} = (A_t \& B_t)$

$A_{t+1} = (\sim A_t \& \sim B_t)$

Part b: Boolean Expressions

$A_{t+1} = (A_t \& B_t) \mid (\sim A_t \& \sim B_t)$

$B_{t+1} = (A_t \& \sim B_t) \mid (\sim A_t \& \sim B_t)$

$= (A_t \mid \sim A_t) \& \sim B_t$

$= \sim B_t$

Question 1: Circuit Design

Part a: Truth Table

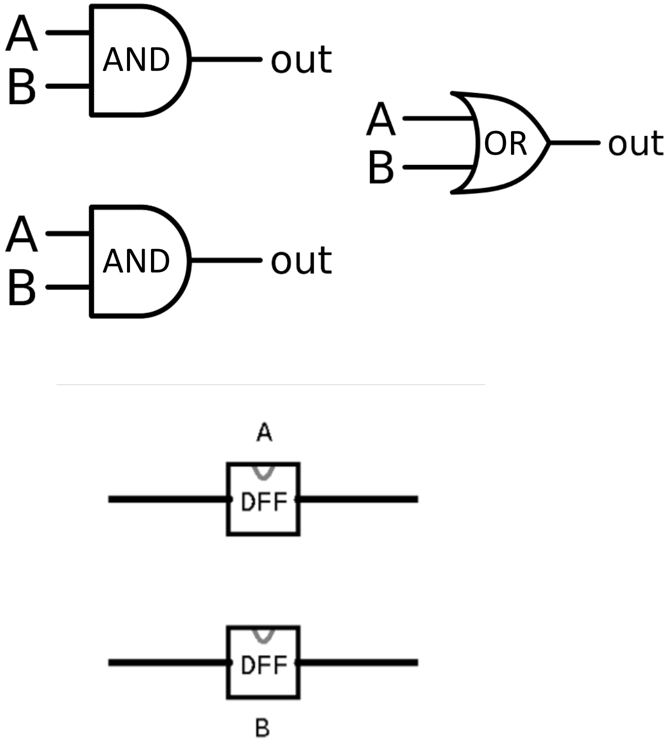
A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

Part b: Boolean Expressions

A_{t+1} = (A_t & B_t) | (~A_t & ~B_t)

B_{t+1} = ~B_t

Part c: Drawing the Circuit



Question 1: Circuit Design

Part a: Truth Table

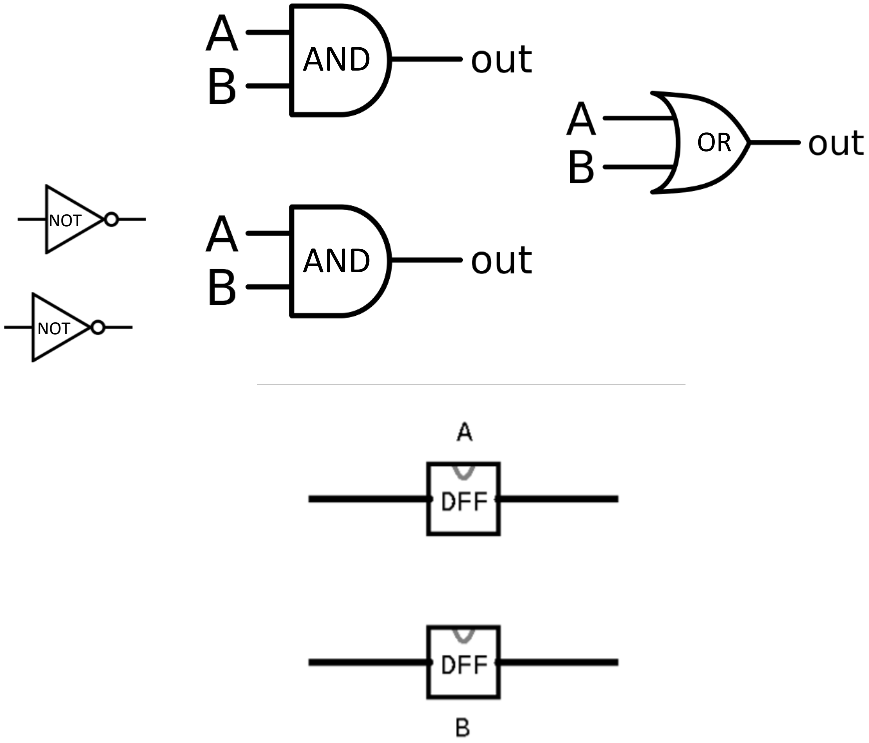
A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

Part b: Boolean Expressions

$A_{t+1} = (A_t \,\&\, B_t) \mid ({\sim}A_t \,\&\, {\sim}B_t)$

$B_{t+1} = {\sim}B_t$

Part c: Drawing the Circuit



Question 1: Circuit Design

Part a: Truth Table

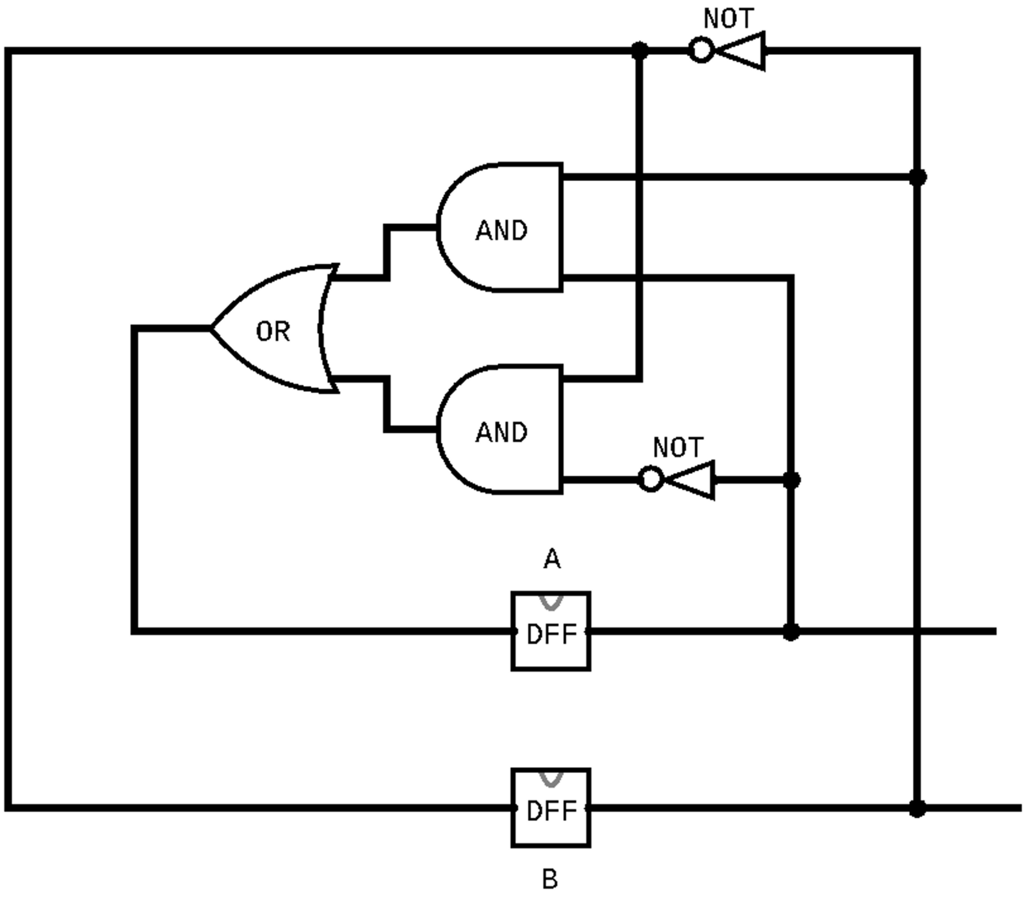
A_t	B_t	->	A_{t+1}	B_{t+1}
1	1		1	0
1	0		0	1
0	1		0	0
0	0		1	1

Part b: Boolean Expressions

$A_{t+1} = (A_t \& B_t) \mid (\sim A_t \& \sim B_t)$

$B_{t+1} = \sim B_t$

Part c: Drawing the Circuit



Question 1: Circuit Design Sample Rubric

Category	Points	Criteria
Truth Table	4 points	1 point for each row in the truth table that is correct
Boolean expressions	6 points	● 4 points for correct expression for A_{t+1} ○ 2 points if truth table is wrong but expression matches truth table ● 2 points for correct expression for B_{t+1} ○ 1 point if truth table is wrong but expression matches truth table
Circuit Diagram	5 points	● 3 points for having circuits that match the boolean expressions in part b ● 2 points for fully correct diagram
Total	**15 points**	

Question 2: Math Puzzle

Dana needs 300 pickets for her colorful picket fence. She wants equal amounts of each of her 4 selected colors. She already has 32 red, 26 green, 9 yellow, and no blue. If the pickets cost 25 cents and you get 20% off if you purchase 50 or more of the same color, and 30% off if you purchase 60 or more of one color, how much does Dana need to spend? List your answer to two decimal places. You may use a calculator application on your computer to solve this problem.

Question 2: Math Puzzle

Dana needs 300 pickets for her colorful picket fence. She wants equal amounts of each of her 4 selected colors. She already has 32 red, 26 green, 9 yellow, and no blue. If the pickets cost 25 cents and you get 20% off if you purchase 50 or more of the same color, and 30% off if you purchase 60 or more of one color, how much does Dana need to spend? List your answer to two decimal places. You may use a calculator application on your computer to solve this problem.

Solution

75 - 32 = 43 red

75 - 26 = 49 green

75 - 9 = 66 yellow

75 - 0 = 75 blue

(rounding down is fine too)

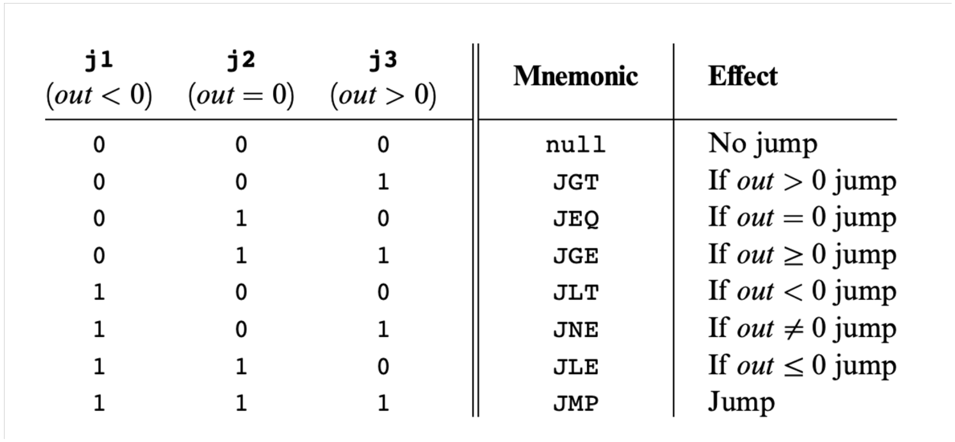
43 * 0.25 + 49 * 0.25 + 0.7 * 66 * 0.25 + 0.7 * 75 * 0.25 = $47.675 = $47.68

Question 3: Hack Assembly Programming

Write a Hack assembly program that stores -1, 0, or 1 in R1 based on the sign of R0. To be more specific, your program should store a -1 in R1 if R0 is negative, a 0 in R1 if R0 is 0, and a 1 in R1 if R0 is positive.

Equivalent pseudocode:

```
if (R0 < 0){
         R1 = -1;
} else if (R0 > 0){
         R1 = 1;
} else{   //R0 == 0
         R1 = 0;
}
```



j1 (*out* < 0)	j2 (*out* = 0)	j3 (*out* > 0)	Mnemonic	Effect
0	0	0	null	No jump
0	0	1	JGT	If *out* > 0 jump
0	1	0	JEQ	If *out* = 0 jump
0	1	1	JGE	If *out* ≥ 0 jump
1	0	0	JLT	If *out* < 0 jump
1	0	1	JNE	If *out* ≠ 0 jump
1	1	0	JLE	If *out* ≤ 0 jump
1	1	1	JMP	Jump

Our solution:

```
    @R0
    D = M
    @NEGATIVE
    D; JLT
    @POSITIVE
    D; JGT
    // R0 == 0 case
    @R1
    M = 0
    @END
    0; JMP
(NEGATIVE)
    // R0 < 0 case
    @R1
    M = -1
    @END
    0; JMP
(POSITIVE)
    // R0 > 0 case
    @R1
    M = 1
(END)
    @END
    0; JMP
```

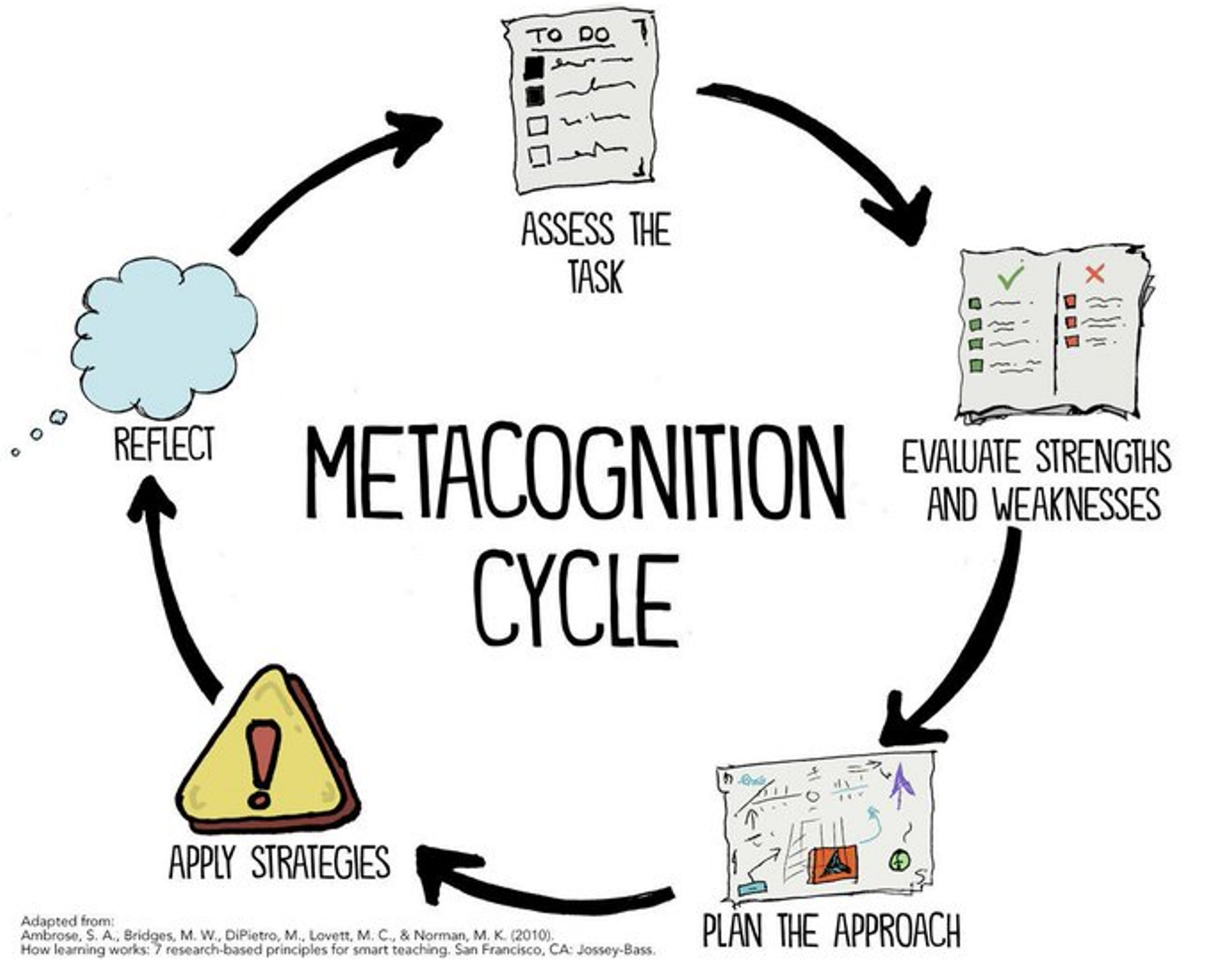
Question 3: Hack Assembly Sample Rubric

Category	Points	Criteria
Has Infinite End Loop	1 point	1 Point if program has an Infinite End Loop
Conditional Checks	4 points	2 points for having at least two checks for cases. Almost all solutions will need a check for 2 of the three cases (negative, zero, positive). 2 points for correctly matching jump condition to cases (e.g. jump to negative case when negative, etc.)
Assigns Correct R1 Value	3 points	One point for each case: negative: R1 = -1 zero: R1 = 0 positive: R = 1
Fully Correct Implementation	2 points	Covers any little mistakes that may result in a not quite correct implementation (e.g. forgetting to jump to the end when a case is done).
Total	**10 points**	

Question 4: Metacognitive Skills

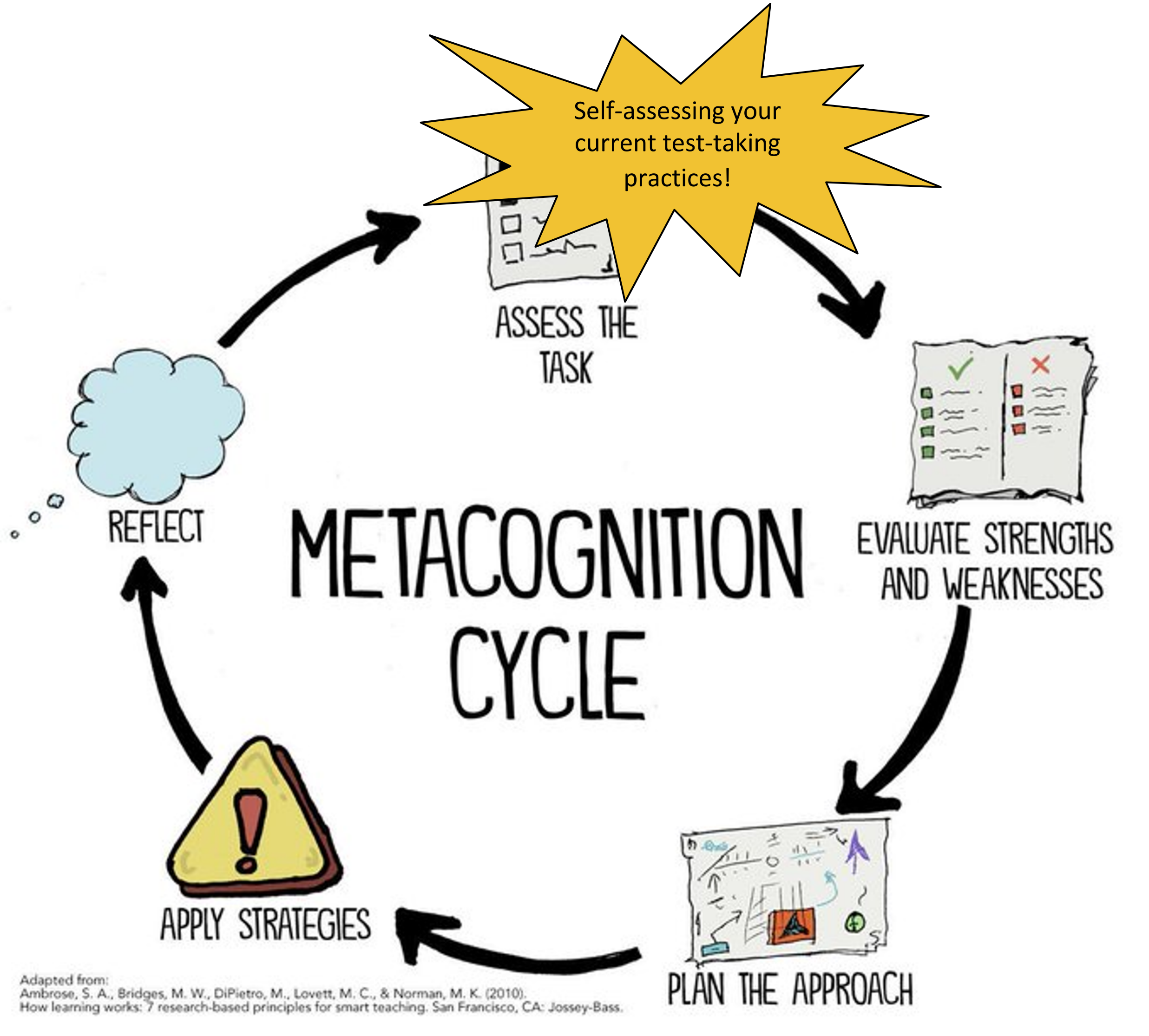
- Name two (2) metacognitive skills that we have covered in CSE 390B thus far.

Why are we making you doing this…



Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

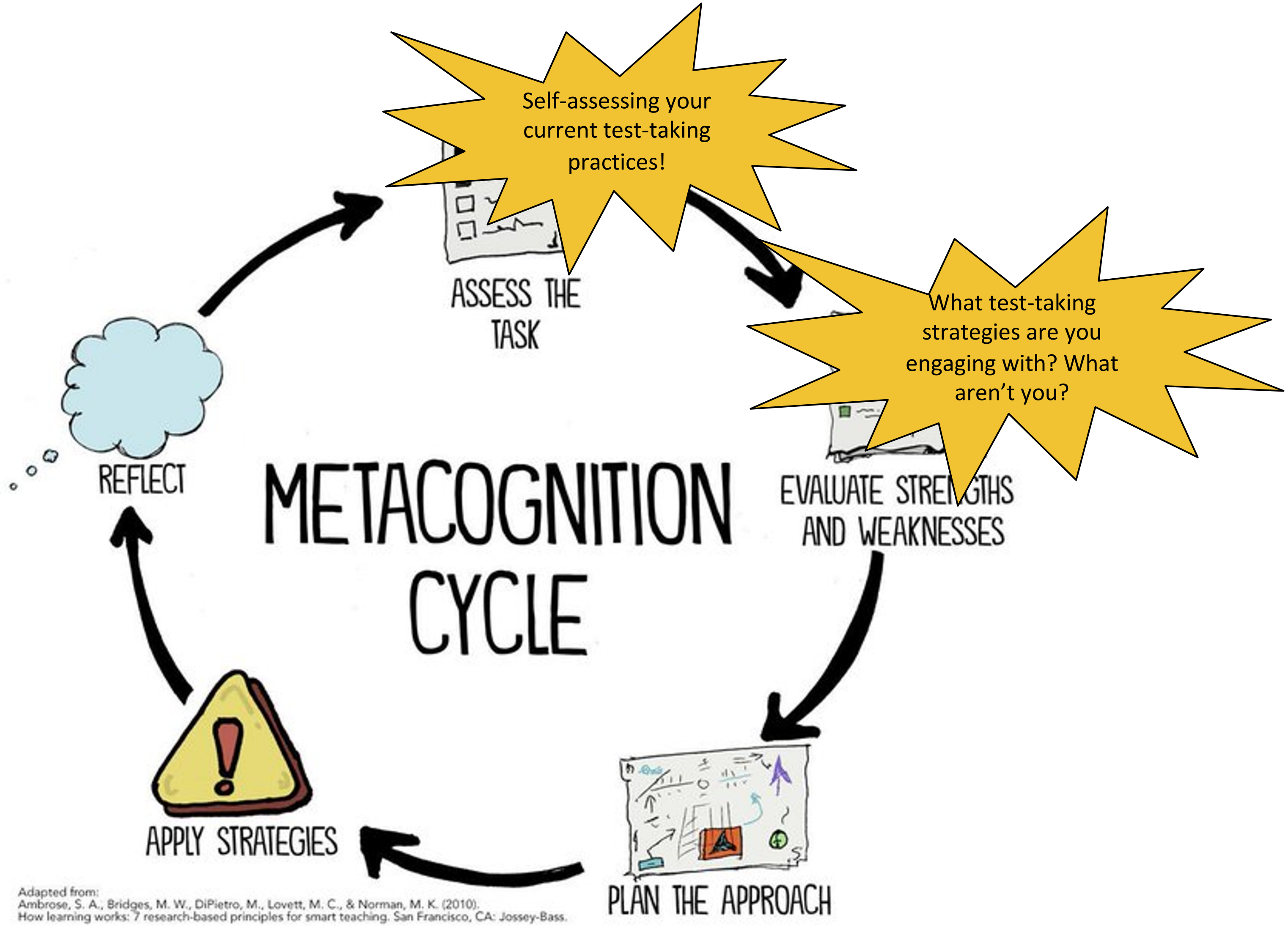
Why are we making you doing this…



Self-assessing your current test-taking practices!

Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Why are we making you doing this…

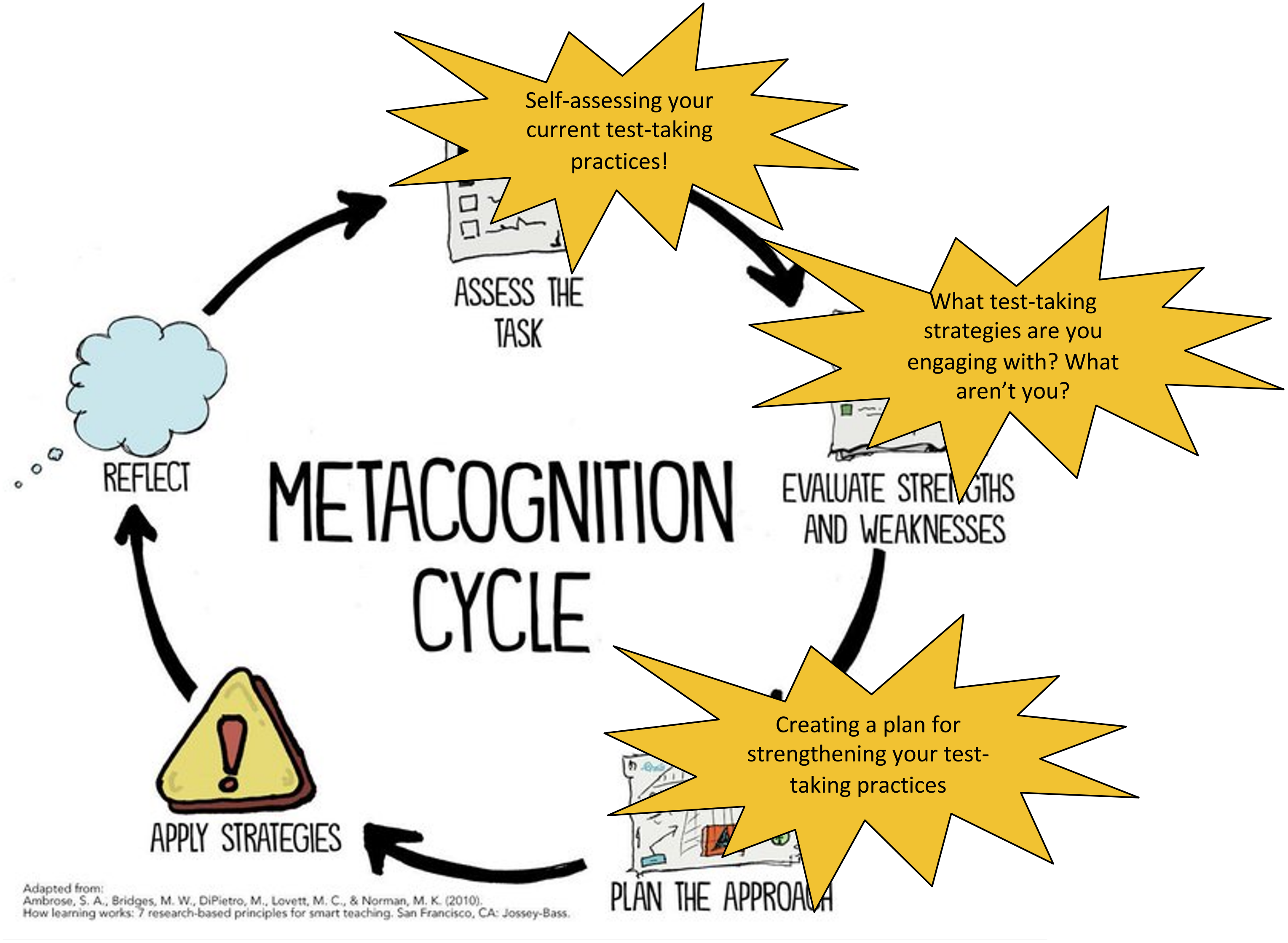


Self-assessing your current test-taking practices!

What test-taking strategies are you engaging with? What aren't you?

Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Why are we making you doing this…



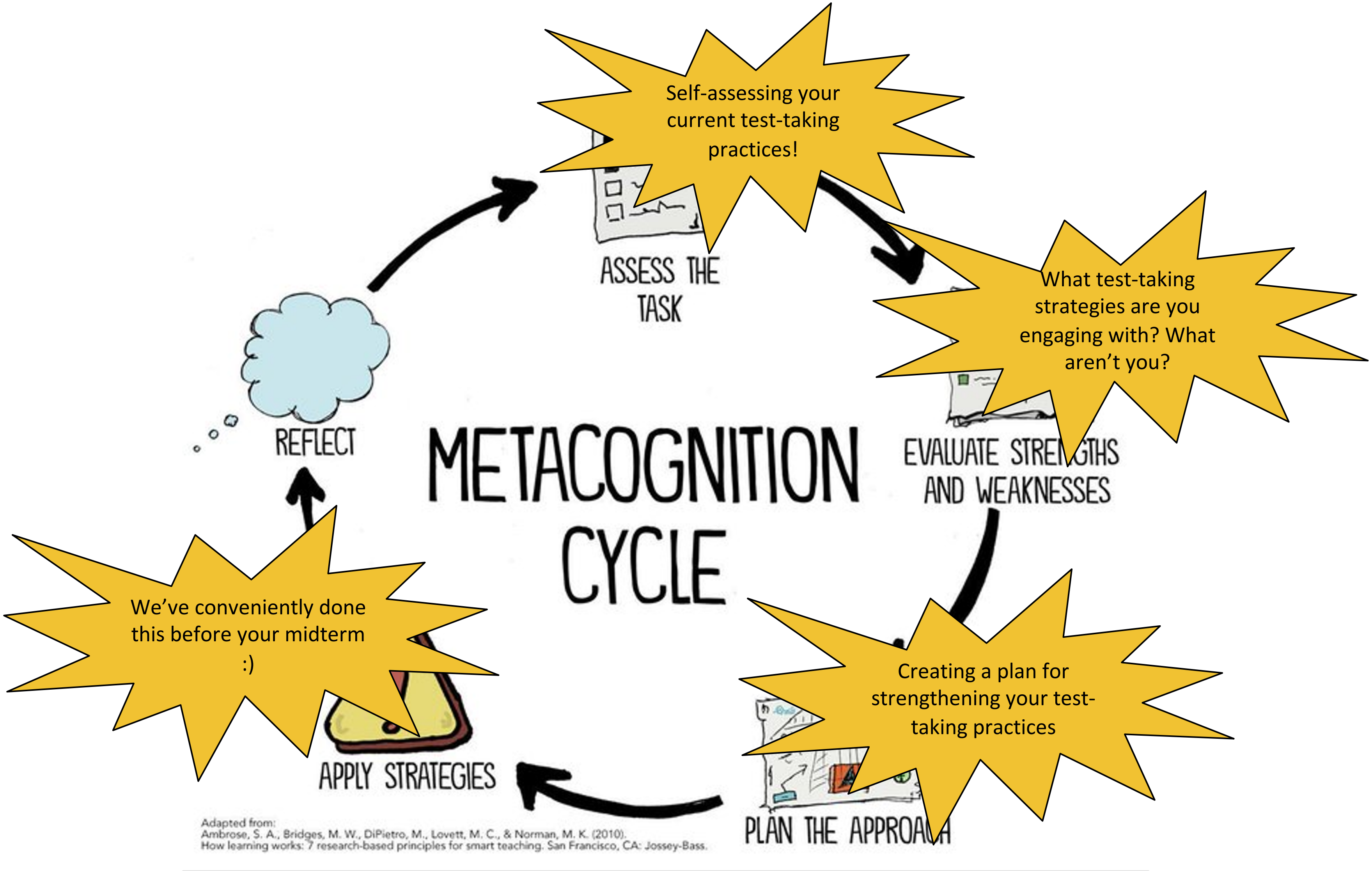
Self-assessing your current test-taking practices!

What test-taking strategies are you engaging with? What aren't you?

Creating a plan for strengthening your test-taking practices

Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Why are we making you doing this…



Self-assessing your current test-taking practices!

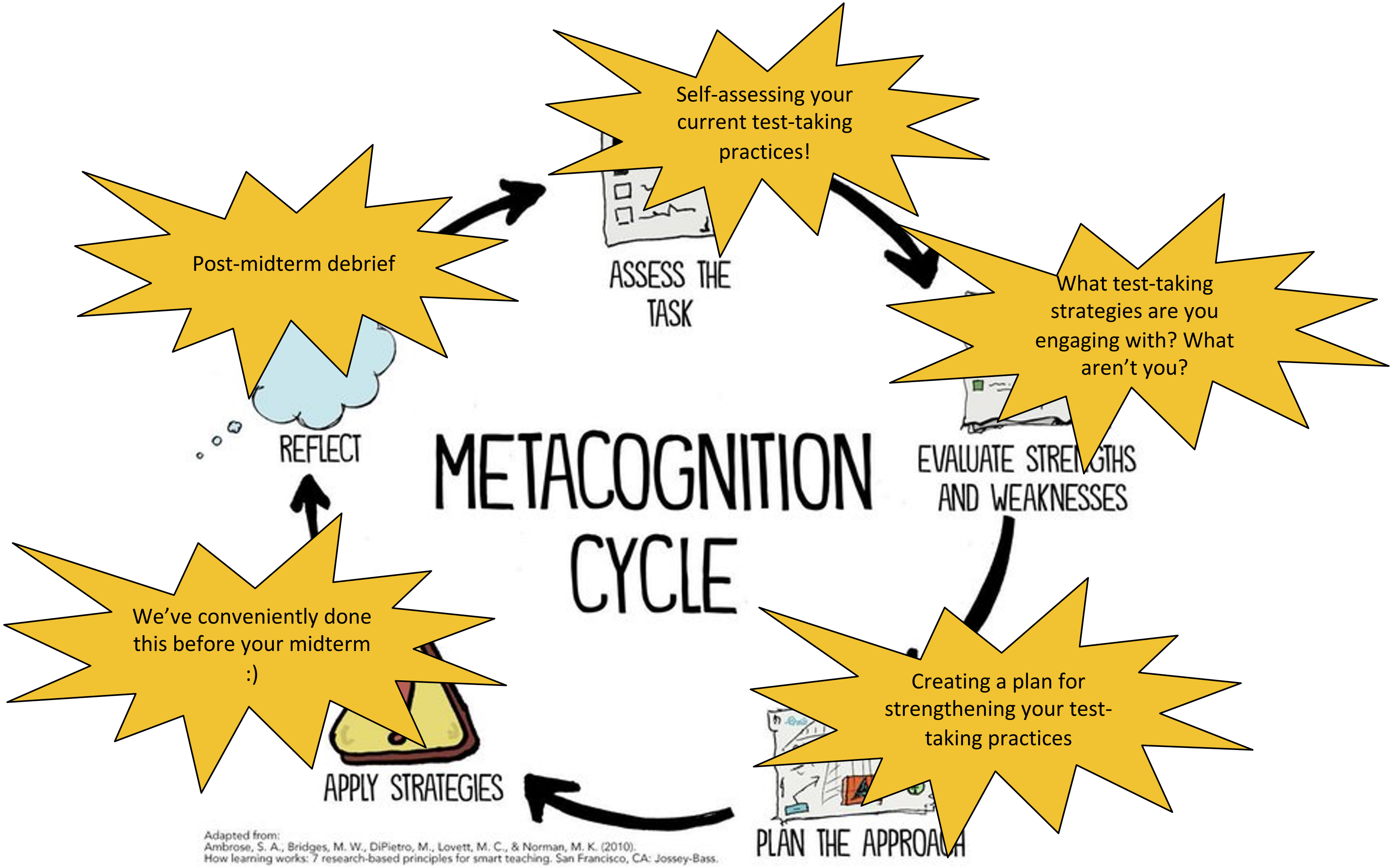
What test-taking strategies are you engaging with? What aren't you?

Creating a plan for strengthening your test-taking practices

We've conveniently done this before your midterm :)

Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Why are we making you doing this...



Self-assessing your current test-taking practices!

What test-taking strategies are you engaging with? What aren’t you?

Creating a plan for strengthening your test-taking practices

We’ve conveniently done this before your midterm :)

Post-midterm debrief

Adapted from:
Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010).
How learning works: 7 research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Previous CSE 390B Midterms

- We have three old midterms from previous quarters
  - Spring 2020 is likely more difficult than the midterm this quarter
  - Winter 2021 & Spring 2021 are more similar to what this quarter’s midterm will look like

- Spring 2020 midterm recommended to become familiar with problem types

- Winter and Spring 2021 midterms recommended for practicing a timed exam
  - Set a timer for 50 minutes and take the exam in its entirety
  - Doing so help you practice time management

Project 5: Overview

- Timed Mock Exam Problem

- Build a Computer
  - `LoadAReg.hdl`, `LoadDReg.hdl` (Easier)
  - `JumpLogic.hdl` (Medium)
  - `CPU.hdl` (Harder)
  - `Computer.hdl` (Easier)

Project 5: Timed Mock Exam Problem

- Your group will meet for a 30-minute session to do one mock exam problem
  - Your group's mock exam problem will be emailed right before your session

- Your 30-min session will include:
  - Set-Up: 5 minutes
  - Mock Exam Problem: 10 minutes
  - Debrief & Reflection: 15 minutes

- Complete and submit the reflection questions

Project 5 Tips

- `CPU.hdl`: We provide an overview diagram, but there are several details to fill in, especially control
  - Crucial to draw your own detailed diagram first
  - Handling jumps will require a lot of logic; sketch out all cases
  - Chapter 4 and 5 are going to be extremely useful

- Multi-Bit Buses: MSB to the left, LSB to the right
  - Important to keep in mind when taking apart the instruction

- Debugging: Consult .out and .cmp files when getting buggy output, then look at internal wires in simulator
  - See also the "Debugging tips" section of the spec

Hack CPU Logic

- How do we determine the unimplemented logic for the CPU (all of the c’s in the diagrams)?

- Need to refer to the assembly specification

- Project 5 will requires understanding of textbook chapters to determine how to use the instruction bits to implement the control logic
  - Textbook sections 4.2.2, 4.2.3, and 5.3.1 are especially helpful

Hack CPU Logic Workflow

- Step 1: What do we pay attention to?
  - Usually, will be some combination of instruction bits or intermediate outputs
  - These are the “inputs” to your sub-problem

- Step 2: Determine logic for the part you are working on
  - Uses the “inputs” from step 1
  - Usually requires reading a relevant section of the textbook/assembly specification

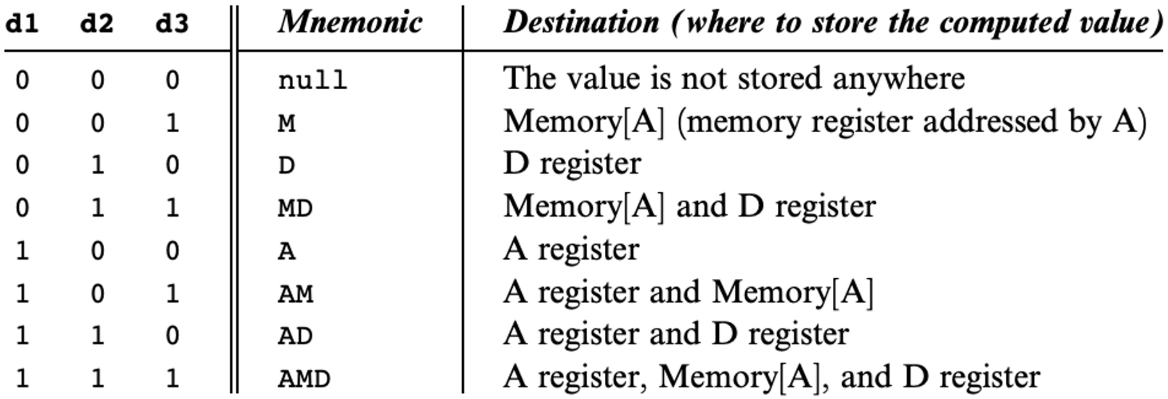
Lecture Outline

- CSE 390B Midterm Brainstorm
- CSE 390B Exam Review Session
  - Circuit Design, Writing Assembly, Tracing Assembly
- Project 5 Overview
  - Timed Mock Exam, Building a Computer, Project Tips
- **Hack CPU Logic Example: writeM**

Hack CPU Logic Example: writeM

- Example: Determine when writeM should be set to 1

- Step 1: What do we pay attention to?
  - **`writeM`** is related to whether we write to memory or not
  - We need to look up the destination bits specification from Chapter 4



d1	d2	d3	*Mnemonic*	*Destination (where to store the computed value)*
0	0	0	null	The value is not stored anywhere
0	0	1	M	Memory[A] (memory register addressed by A)
0	1	0	D	D register
0	1	1	MD	Memory[A] and D register
1	0	0	A	A register
1	0	1	AM	A register and Memory[A]
1	1	0	AD	A register and D register
1	1	1	AMD	A register, Memory[A], and D register

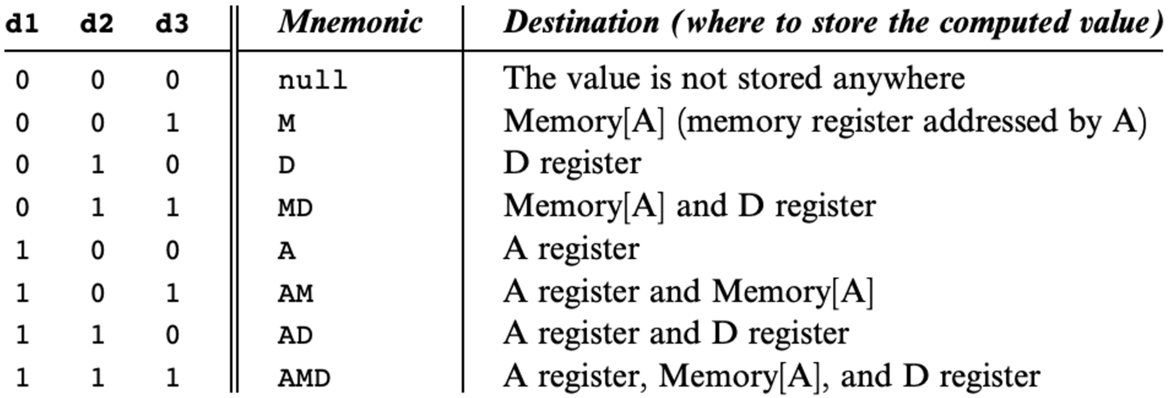
Figure 4.4 The *dest* field of the *C*-instruction.

Hack CPU Logic Example: writeM

- Example: Determine when writeM should be set to 1

- Step 2: Determine logic for specification
  - Read the “Destination Specification” section of Chapter 4
  - Instruction bits:

```
1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3
```



d1	d2	d3	*Mnemonic*	*Destination (where to store the computed value)*
0	0	0	null	The value is not stored anywhere
0	0	1	M	Memory[A] (memory register addressed by A)
0	1	0	D	D register
0	1	1	MD	Memory[A] and D register
1	0	0	A	A register
1	0	1	AM	A register and Memory[A]
1	1	0	AD	A register and D register
1	1	1	AMD	A register, Memory[A], and D register

Figure 4.4 The *dest* field of the *C*-instruction.

Hack CPU Implementation: Logic sub-chips

- We provide you with 3 sub-chips and tests that implement the control logic for the A Register, D Register, and PC
  - `LoadAReg` contains logic for loading the A Register
  - `LoadDReg` contains logic for loading the D Register
  - `JumpLogic` contains logic for determining if the PC should load, jump, or increment

- Implement and test these first, then use them in your CPU implementation
  - Intended to help you narrow the scope of bugs

Post-Lecture 11 Reminders

- CSE 390B Midterm Exam this Thursday (2/10) during in-person lecture at 1:30pm

- Project 5: Building a Computer Part II and Timed Mocked Exam due on 2/17 at 11:59pm PST
  - Project 4 feedback released thi

- Questions on midterm logistics of concepts or Project 5?
  - Eric’s OH: Tue. 3-4pm (hybrid), Wed. 4:30-5:30pm (virtual)
  - Leslie’s OH: Wed. 4:30-5pm (hybrid)
  - Audrey and Sean’s OH: Wed. 1:30-2:30pm (hybrid)
  - Ask questions on the Ed discussion board