CSE 390B, Spring 2023

Building Academic Success Through Bottom-Up Computing

Hack CPU Logic & Midterm Practice

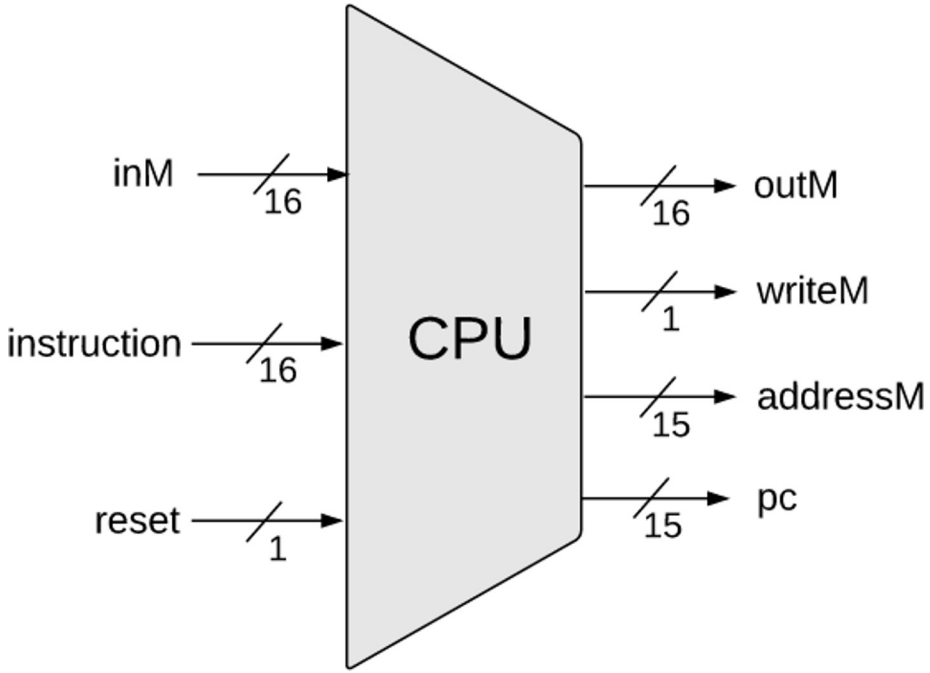
Hack CPU Logic, Midterm Topics Brainstorm and Practice Problems, Project 6 Overview

Lecture Outline

- **Hack CPU Logic**
  - **Implementation and Operations**
- CSE 390B Midterm Practice Problems
  - Midterm Topics Brainstorming
- Project 6 Overview
  - Project Tips and Workflow

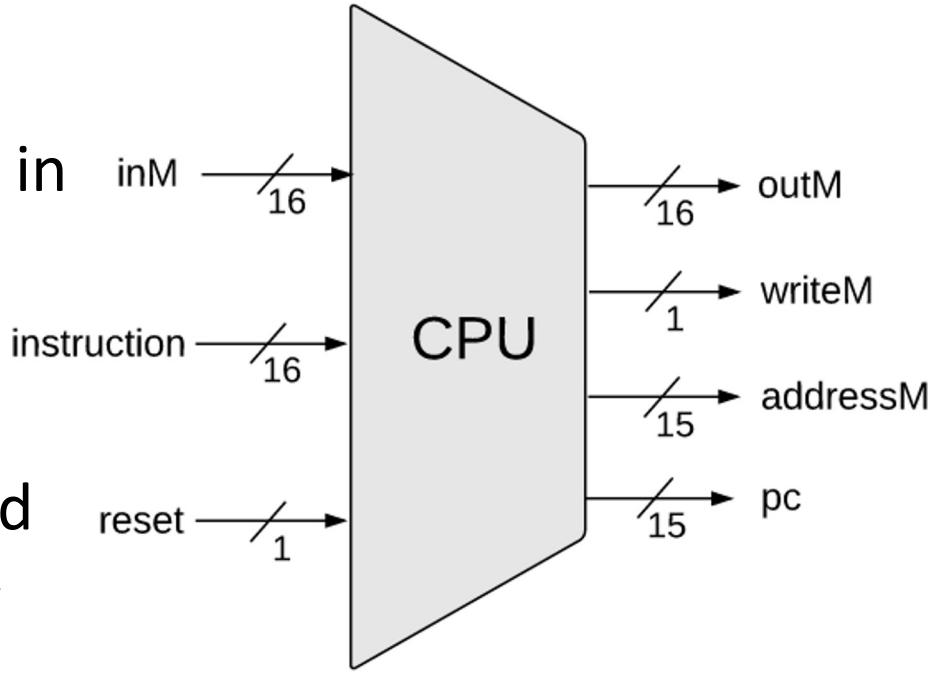
Hack CPU Interface Inputs

- **`inM`**: Value coming from memory
- **`instruction`**: 16-bit instruction
- **`reset`**: if 1, reset the program

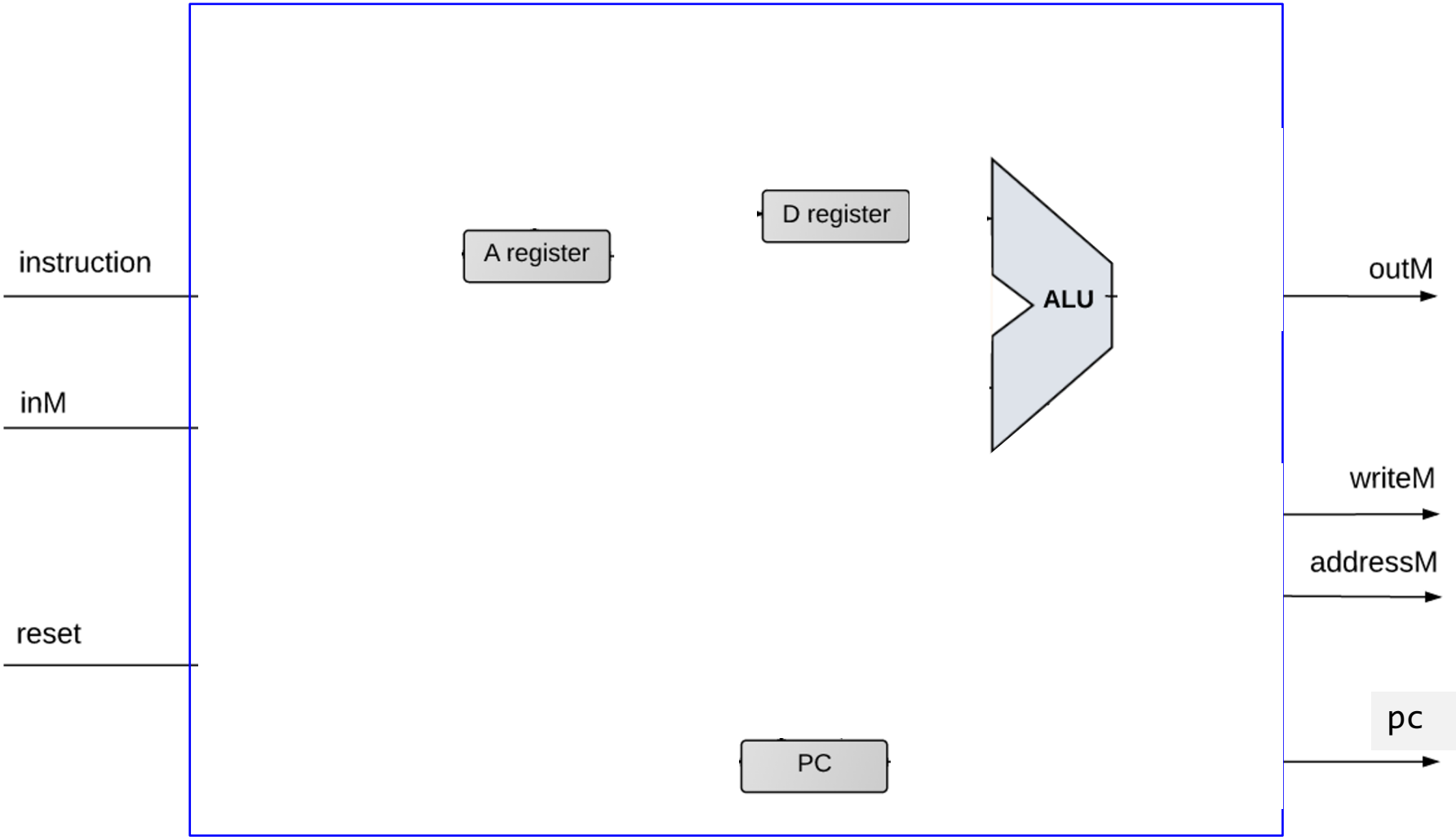


Hack CPU Interface Outputs

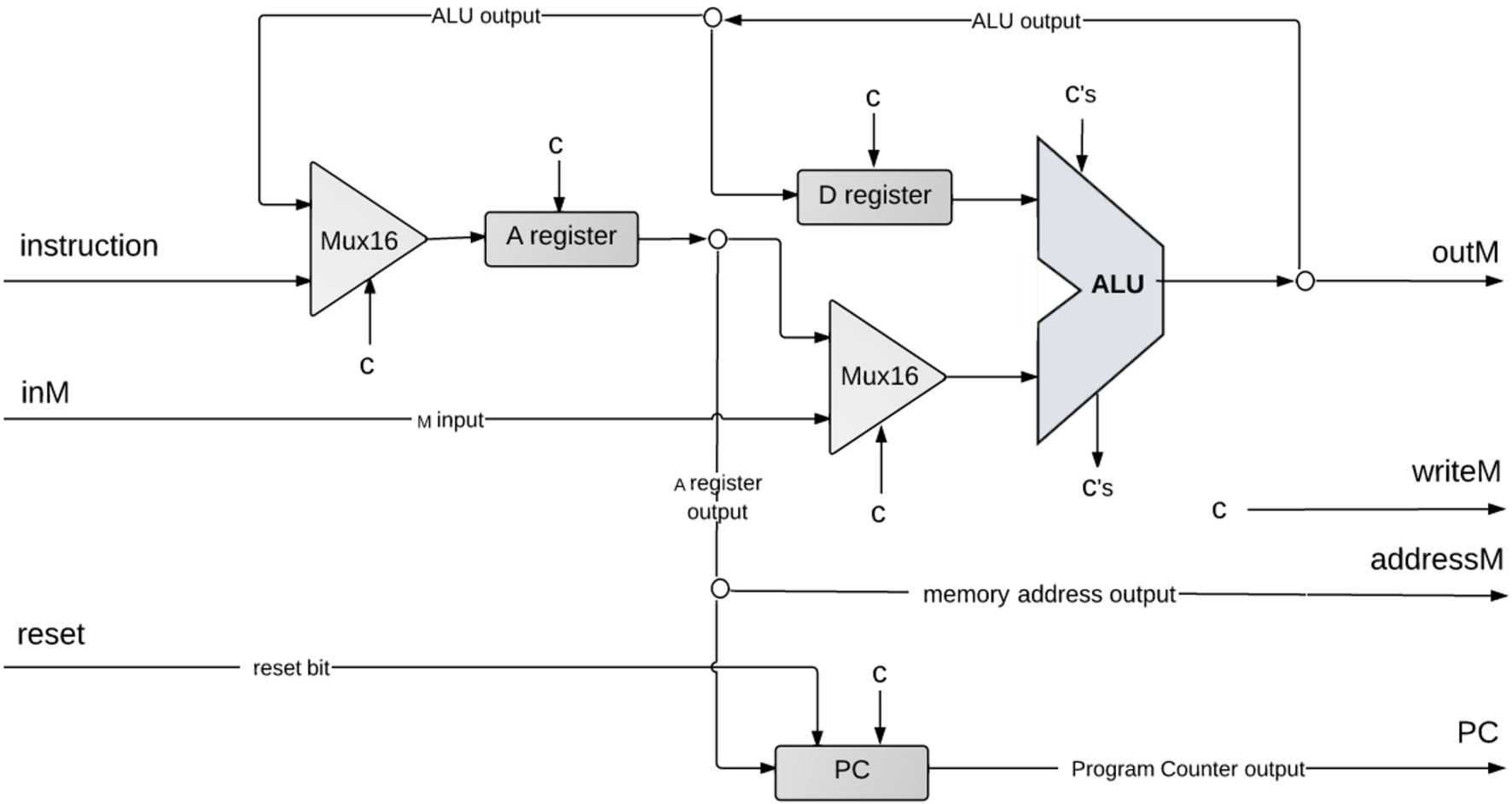
- **`outM`**: value used to update memory if writeM is 1
- **`writeM`**: if 1, update value in memory at addressM with outM
- **`addressM`**: address to read from or write to in memory
- **`pc`**: address of next instruction to be fetched from memory



Hack CPU Implementation

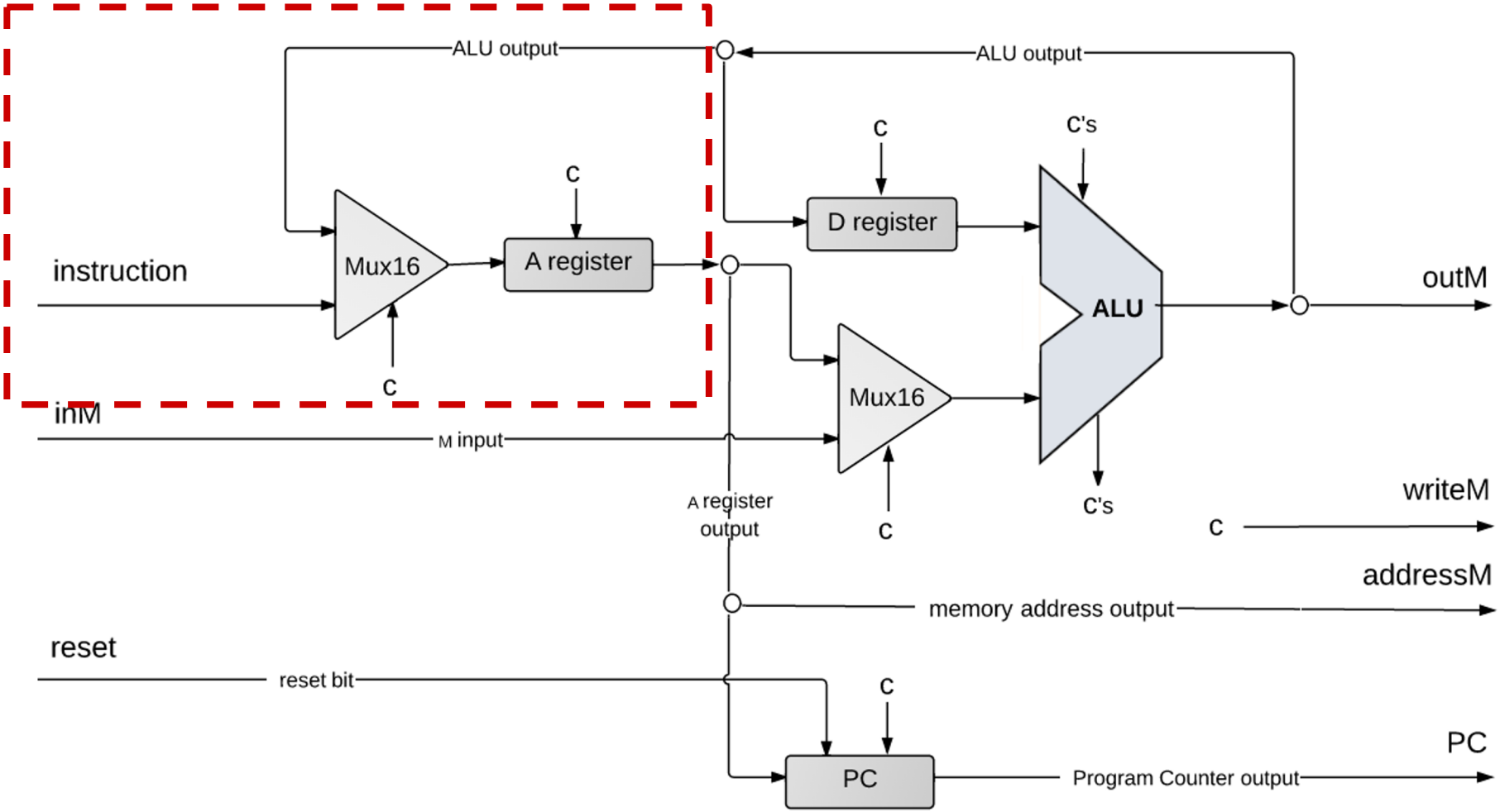


Hack CPU Implementation



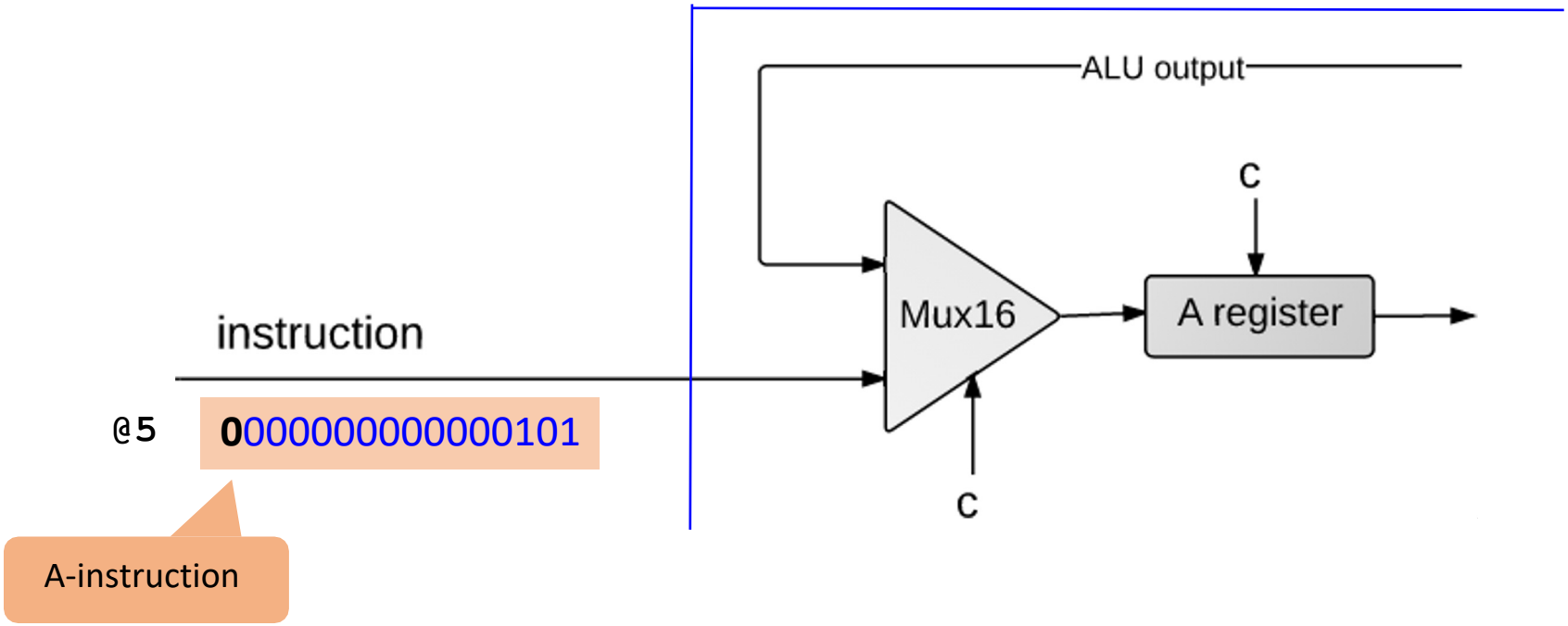
(each "C" symbol represents a control bit)

CPU Operation: Instruction Handling

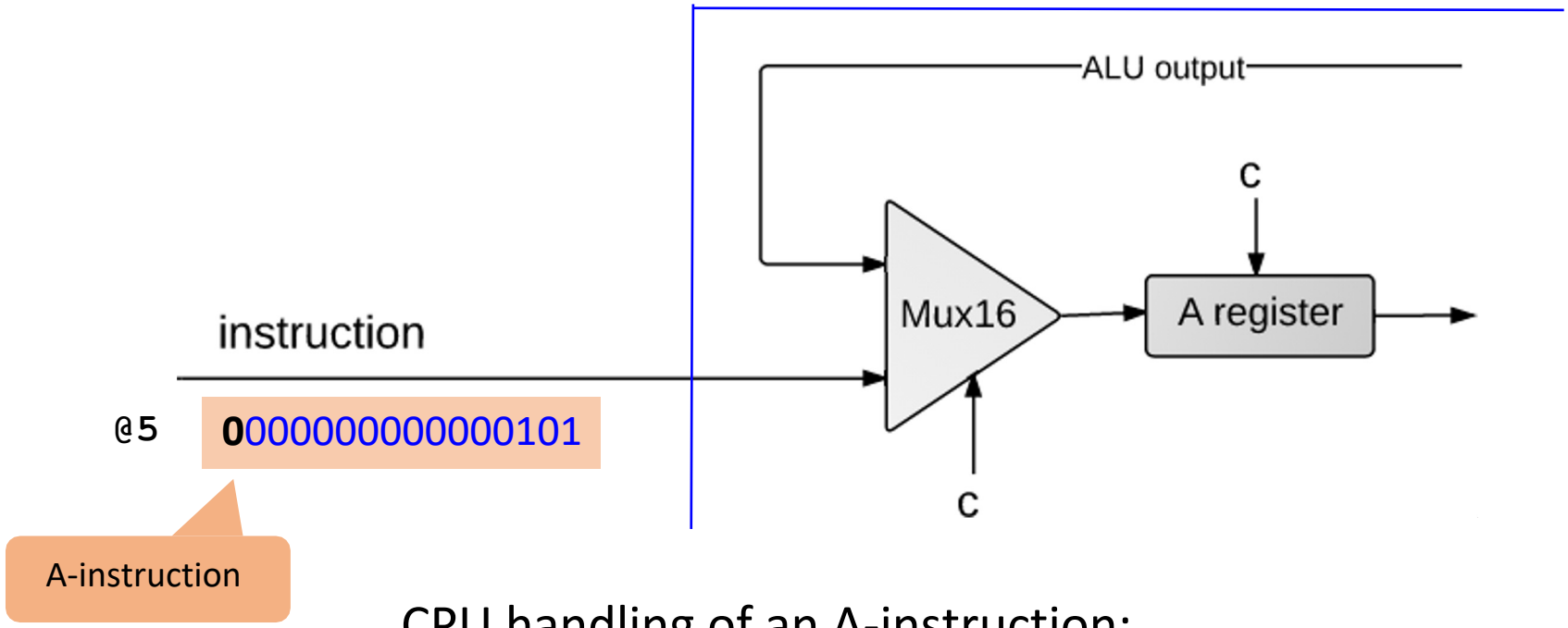


(each "C" symbol represents a control bit)

CPU Operation: Instruction Handling



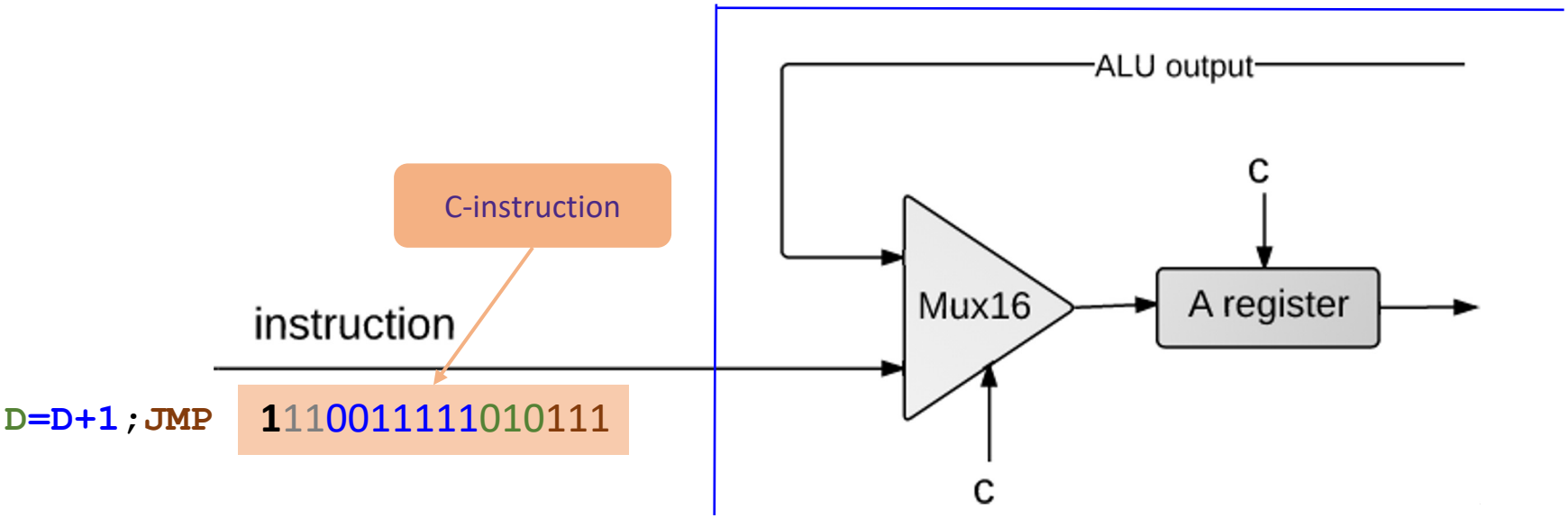
CPU Operation: Instruction Handling



CPU handling of an A-instruction:

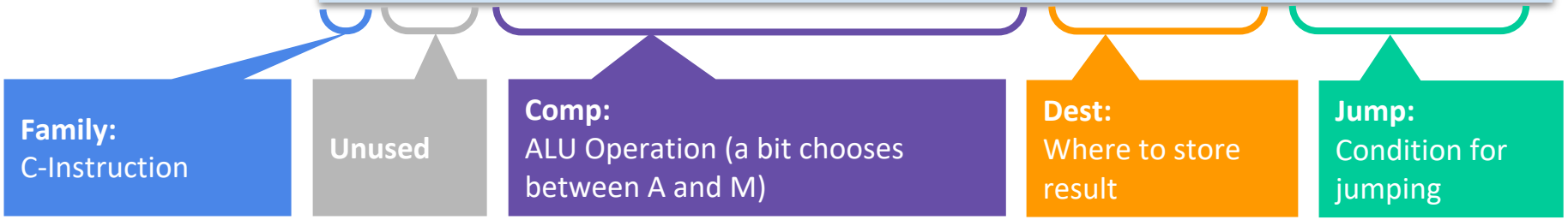
- Decodes the instruction into:
  - op-code
  - 15-bit value
- Stores the value in the A-register
- Outputs the value (not shown in this diagram)

CPU Operation: Instruction Handling

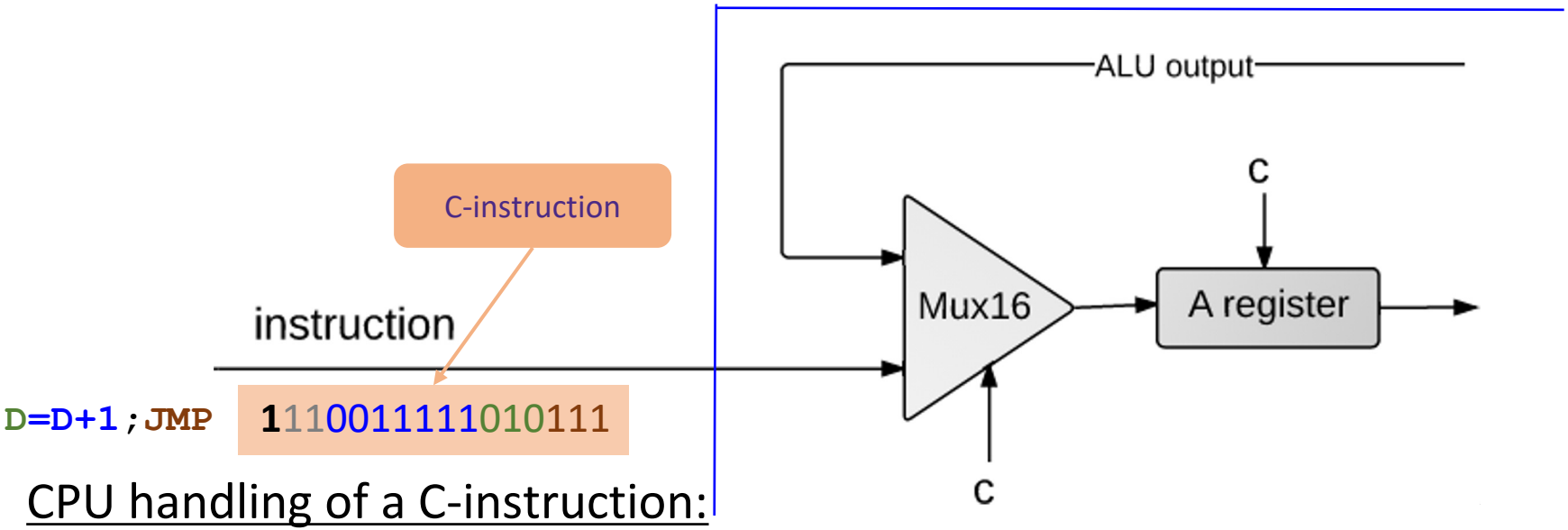


Hack: C-Instructions

- Symbolic: `dest = comp ; jump`
- Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`



CPU Operation: Instruction Handling



CPU handling of a C-instruction:

- Decodes the instruction bits into:
  - Op-code
  - ALU control bits
  - Destination load bits
  - Jump bits
- Routes these bits to their chip-part destinations
- The chip-parts (most notably, the ALU) execute the instruction

Hack: C-Instructions

- Symbolic: `dest = comp ; jump`
- Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

Comp:
ALU Operation (a bit chooses between A and M)

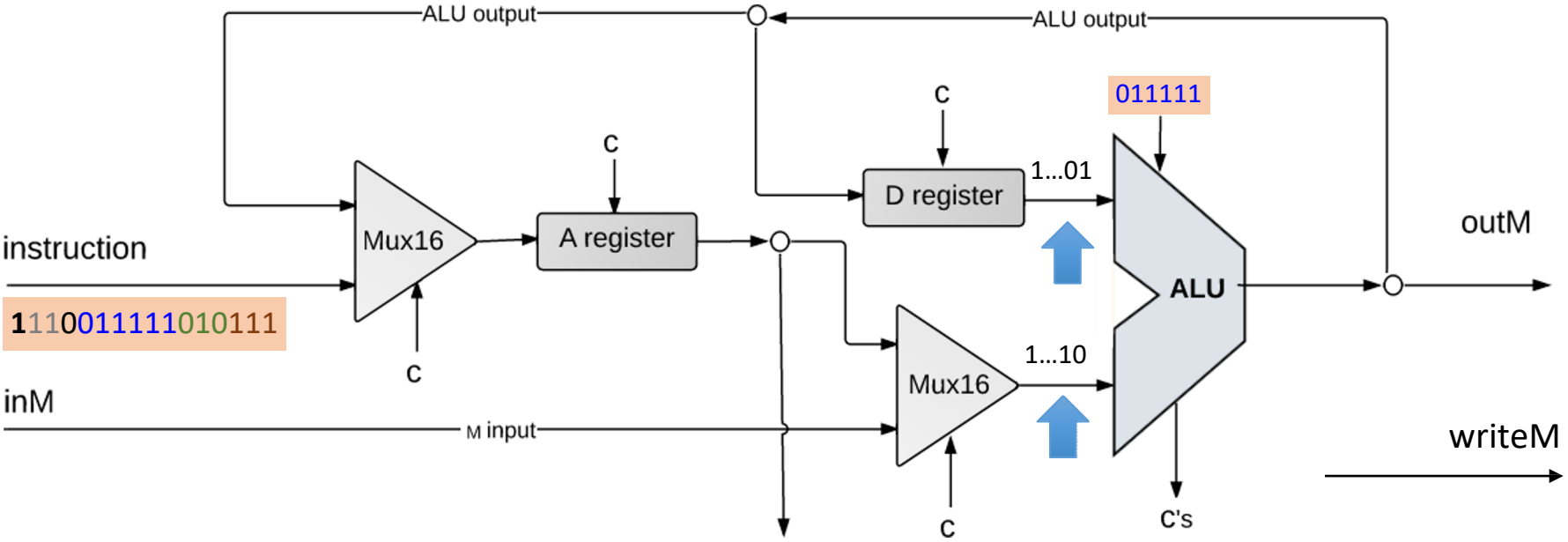
(when a=0) *comp mnemonic*	c1	c2	c3	c4	c5	c6	(when a=1) *comp mnemonic*
0	1	0	1	0	1	0	
1	1	1	1	1	1	1	
-1	1	1	1	0	1	0	
D	0	0	1	1	0	0	
A	1	1	0	0	0	0	M
!D	0	0	1	1	0	1	
!A	1	1	0	0	0	1	!M
-D	0	0	1	1	1	1	
-A	1	1	0	0	1	1	-M
D+1	0	1	1	1	1	1	
A+1	1	1	0	1	1	1	M+1
D-1	0	0	1	1	1	0	
A-1	1	1	0	0	1	0	M-1
D+A	0	0	0	0	1	0	D+M
D-A	0	1	0	0	1	1	D-M
A-D	0	0	0	1	1	1	M-D
D&A	0	0	0	0	0	0	D&M
D|A	0	1	0	1	0	1	D|M

Chapter 4

Important: just pattern matching text!

Cannot have "**1+M**"

CPU Operation: Handling C-Instructions



ALU data inputs:

- Input 1: from the D-register
- Input 2: from either:
  - A-register, or
  - data memory

ALU control inputs:

- Control bits (from the instruction)

Hack: C-Instructions

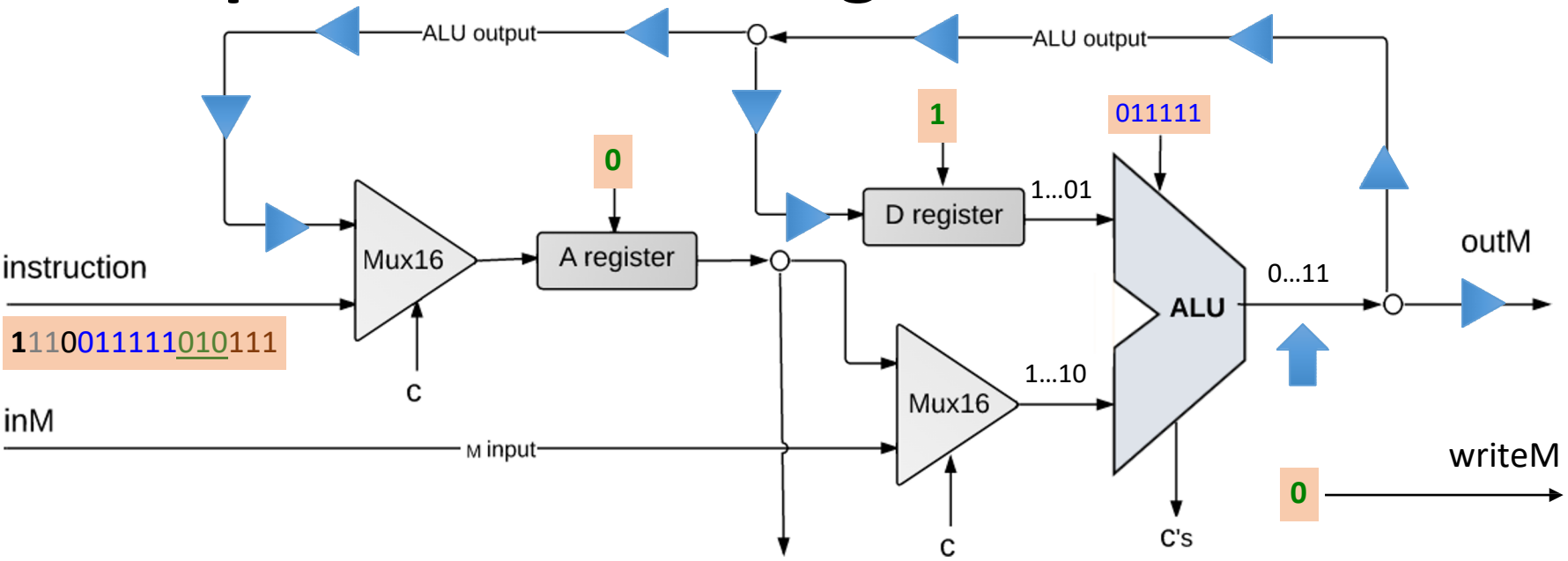
- Symbolic: `dest = comp ; jump`
- Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

Dest:
Where to store result

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

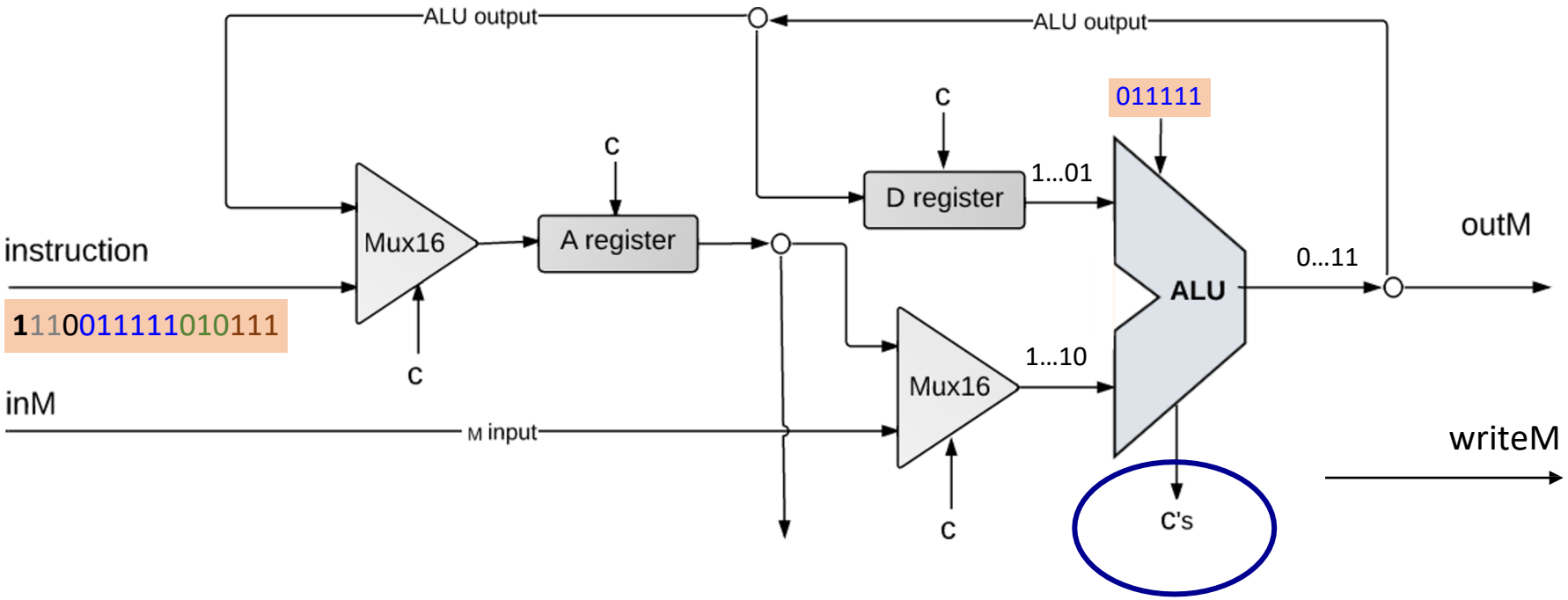
CPU Operation: Handling C-Instructions



ALU data output:

- Result of ALU calculation
- Fed simultaneously to: D-register, A-register, data memory
- Which destination *actually* commits to the ALU output is determined by the instruction's destination bits

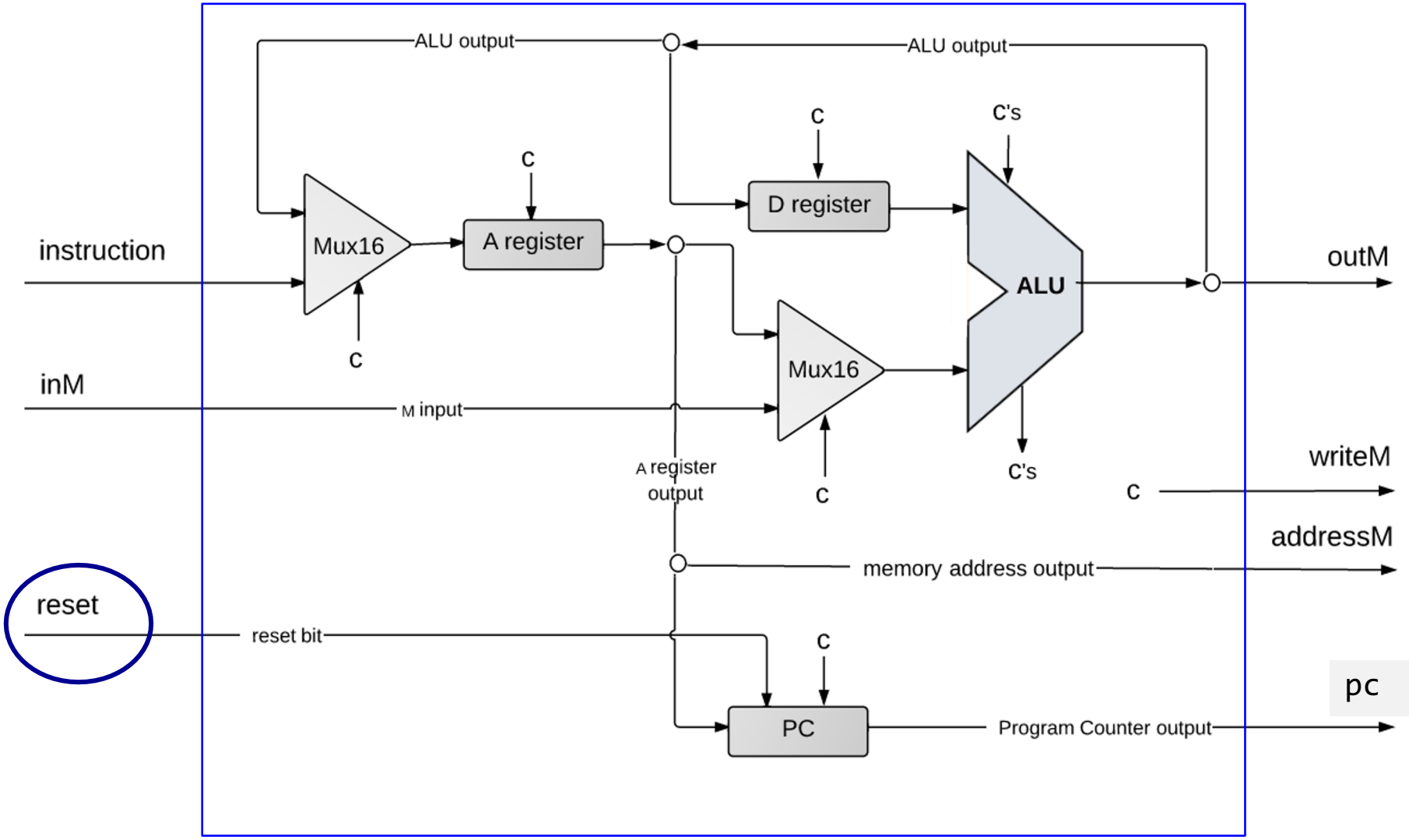
CPU Operation: Handling C-Instructions



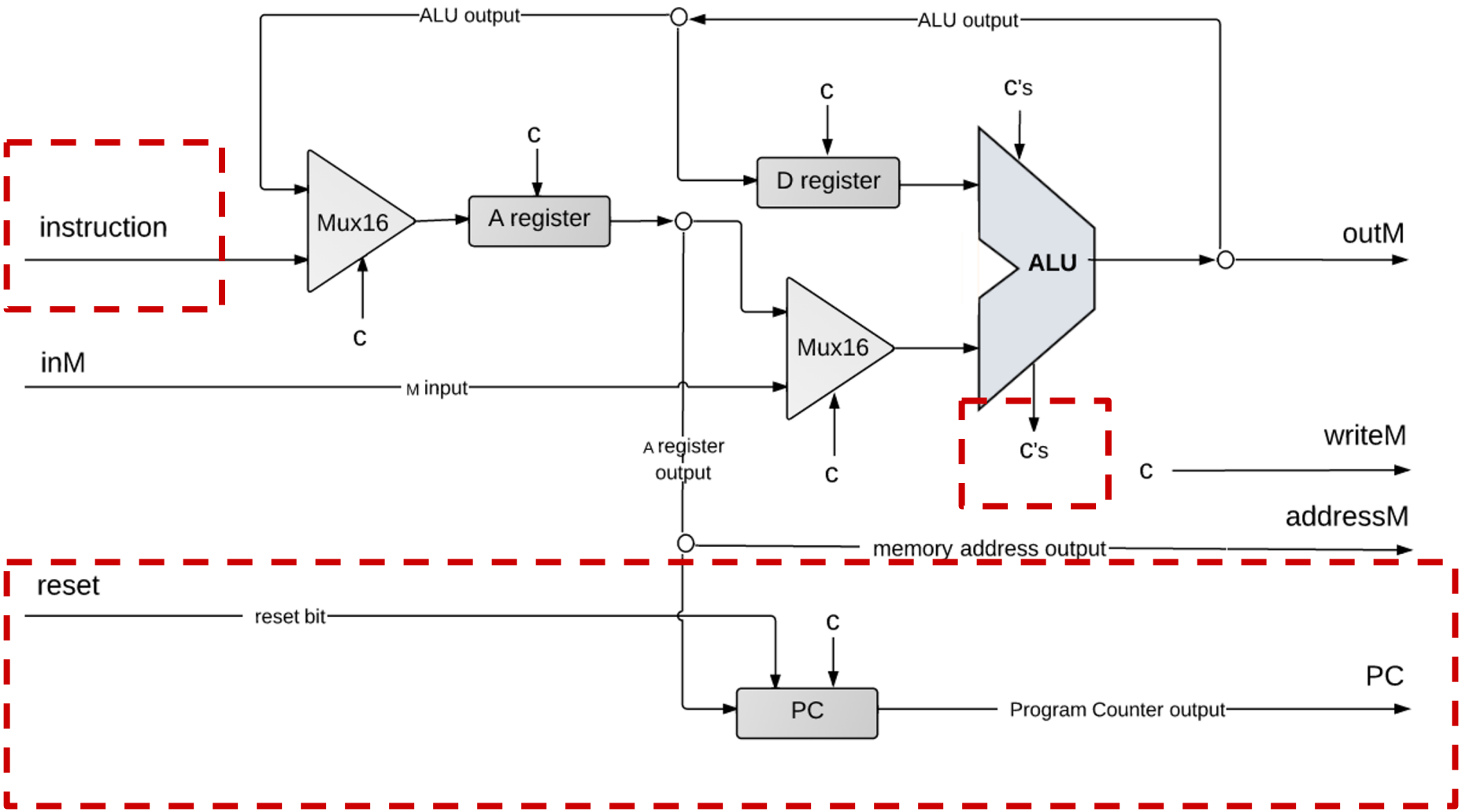
ALU control outputs:

- Is the output negative?
- Is the output zero?

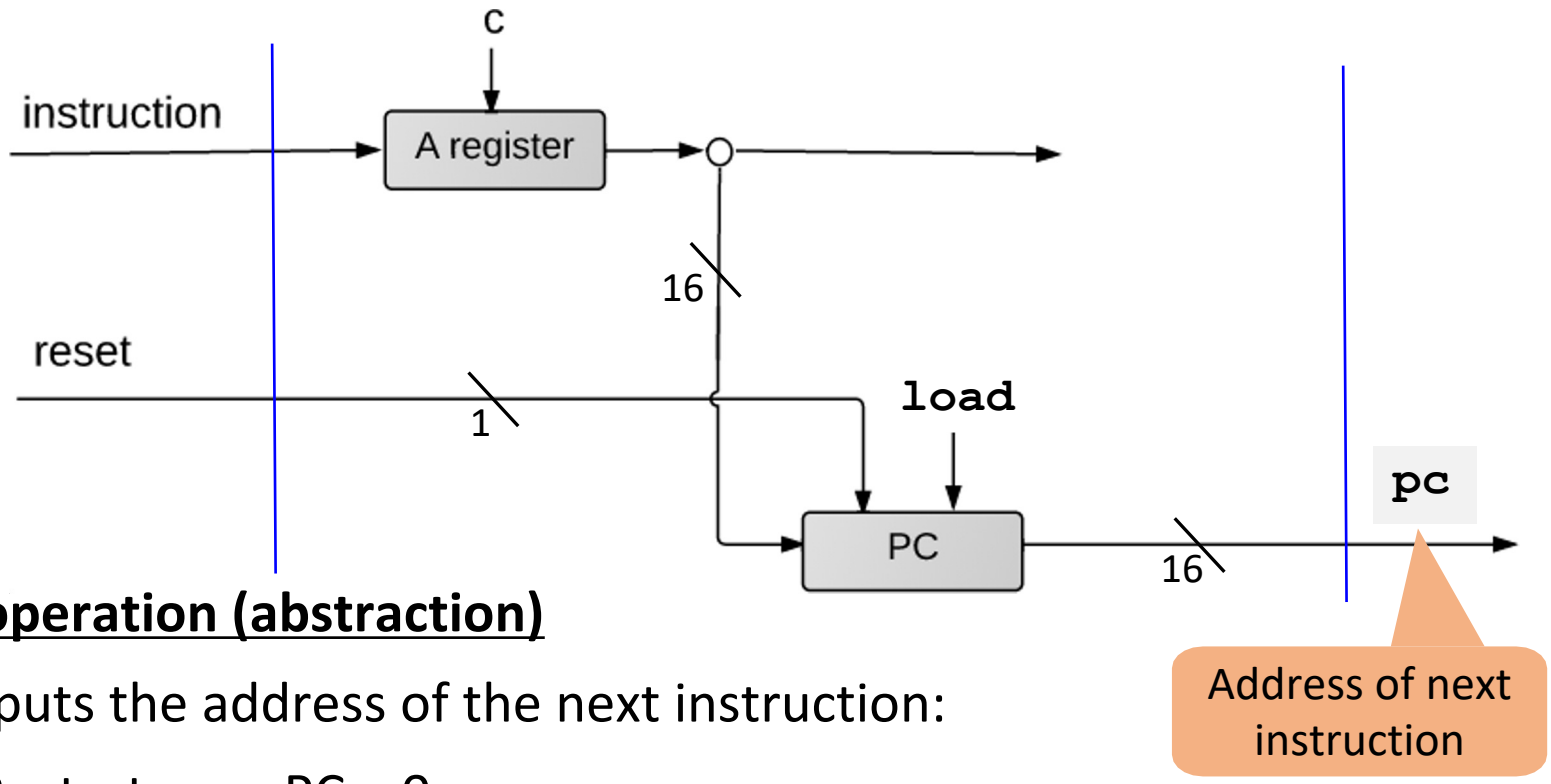
CPU Operation: Control



CPU Operation: Control



CPU Operation: Control



PC operation (abstraction)

Outputs the address of the next instruction:

- Restart: PC = 0
- No jump: PC++
- Go to: PC = A
- Conditional go to: if (condition) PC = A
  else PC ++

Hack: C-Instructions

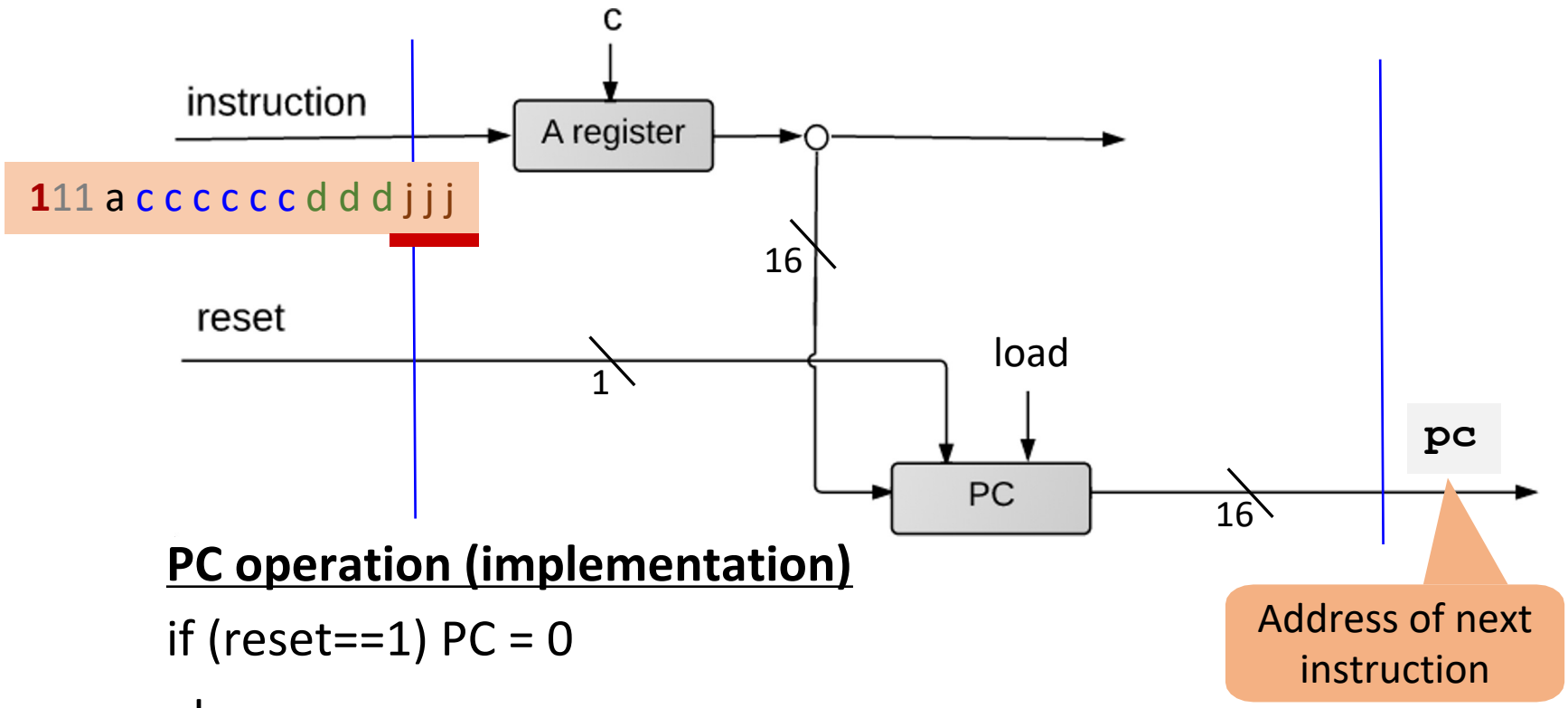
- Symbolic: `dest = comp ; jump`
- Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

Jump: Condition for jumping

Chapter 4

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 $out \geq 0$ jump
1	0	0	JLT	If $out < 0$ jump
1	0	1	JNE	If $out \neq 0$ jump
1	1	0	JLE	If $out \leq 0$ jump
1	1	1	JMP	Jump

CPU Operation: Control



PC operation (implementation)

if (reset==1) PC = 0

else

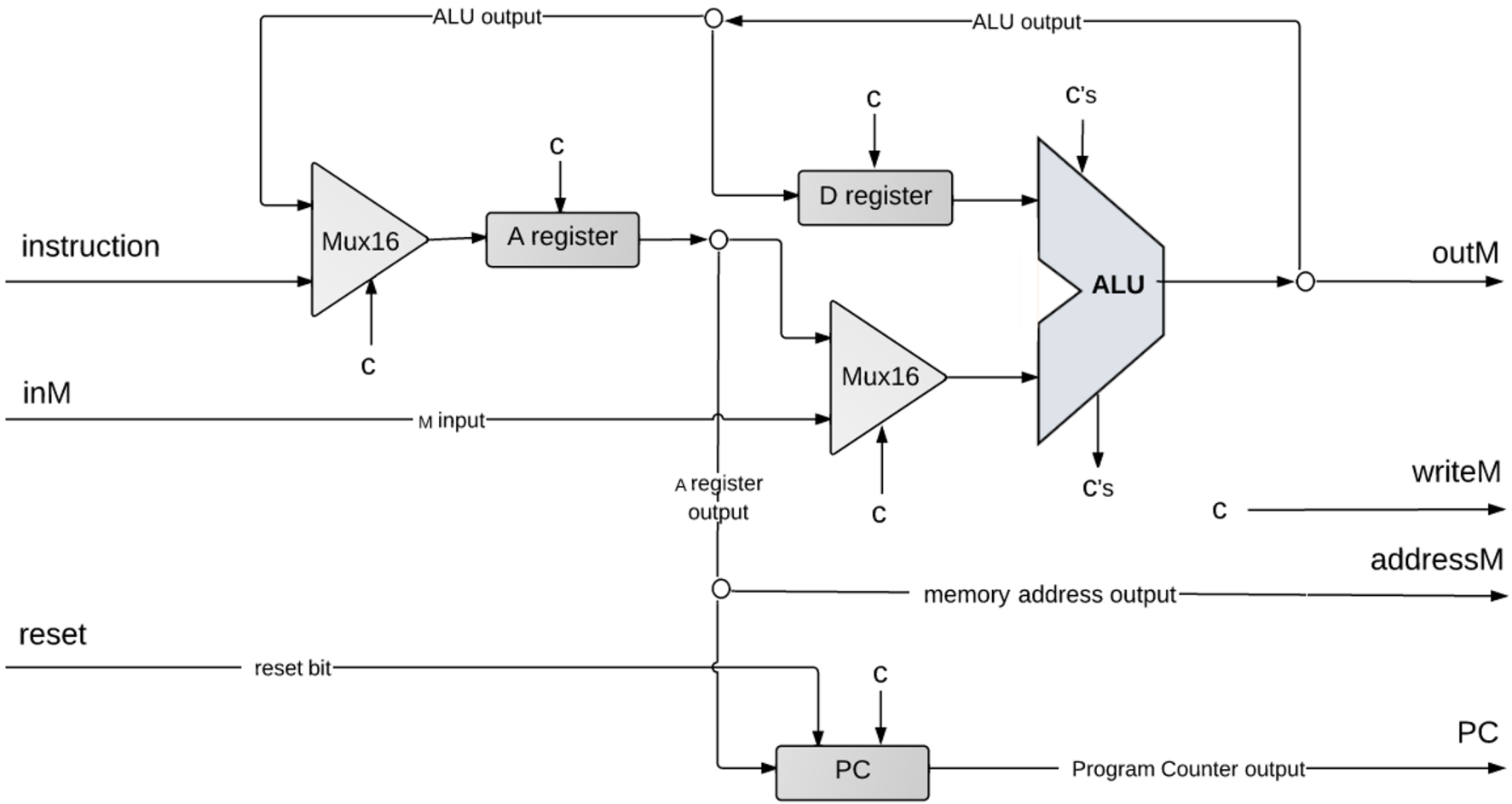
// In the course of handling the current instruction:

load = *f* (jump bits, ALU control outputs)

if (load == 1) PC = A // jump

else PC++ // next instruction

Hack CPU Implementation: That's It!



Lecture Outline

- Hack CPU Logic
  - Implementation and Operations
- **CSE 390B Midterm Practice Problems**
  - **Midterm Topics Brainstorming**
- Project 6 Overview
  - Project Tips and Workflow

CSE 390B Midterm Topics Brainstorm

- Based on what we have covered thus far in class, what are topics, concepts, questions that you might expect to show up on next week's CSE 390B midterm?

CSE 390B Review Session

- Practice Problems
  - Circuit Design, Writing Assembly, Tracing Assembly

- For each problem:
  - Step 1: Spend 10 minutes working on the problem individually
  - Step 2: Spend 10 minutes discussing the problem as a group, describing tips, approaches, and test-taking strategies
  - Step 3: As a group, present to the class your discussion from step 2 and lead the class in working through the problem

Review Session Debrief

- Based on your experience with this exercise, how does it inform how you approach your studying?

- What resources can you utilize to help you deepen your understanding?

Previous CSE 390B Midterms

- Four midterms from previous quarters
  - 20sp midterm likely more difficult than midterm this quarter
  - Midterms from 21wi, 21sp, 22wi, 22sp, 22au, and 23wi are more like what this quarter's midterm will look like
- 20sp midterm recommended to become familiar with problem types
- 21wi, 21sp, 22wi, 22sp, 22au, and 23wi midterms recommended for practicing a timed exam
  - Set a timer for 60 minutes and take the exam in its entirety
  - Helps practice time management and simulate exam environment

Lecture Outline

- Hack CPU Logic
  - Implementation and Operations
- CSE 390B Midterm Practice Problems
  - Midterm Topics Brainstorming
- **Project 6 Overview**
  - **Project Tips and Workflow**

Project 6: Overview

- Part I: Mock Exam Problem

- Part II: Building a Computer
  - **`LoadAReg.hdl`**, **`LoadDReg.hdl`** (Easier)
  - **`JumpLogic.hdl`** (Medium)
  - **`CPU.hdl`** (Harder)
  - **`Computer.hdl`** (Easier)

- Part III: Project 6 Reflection

Project 6, Part I: 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-minute session will include:
  - Set up: 5 minutes
  - Mock Exam Problem: 10 minutes
  - Debrief & Reflection: 15 minutes
- Part I task: Submit the completed mock exam problem and complete the reflection questions

Project 6 Tips

- **`CPU.hdl`**: We provide an overview diagram, but there are details to fill in, especially control
  - Draw your own detailed diagram first
  - Handling jumps will require a lot of logic—sketch out the cases
  - Textbook chapter 4 and 5 helpful for Project 6

- 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 to debug, then look at internal wires in simulator
  - See also the "Debugging tips" section of the specification

Post-Lecture 10 Reminders

- **Project 5 due tonight (4/27) at 11:59pm**
- **CSE 390B midterm next Thursday (5/4) during lecture**
- Project 6 (Mock Exam Problem & Building a Computer) released today, due in two Thursdays (5/11) at 11:59pm
- Anam has office hours after class in CSE2 121
  - Feel free to post your questions on the Ed board as well