

CSE 390B, 2024 Winter

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

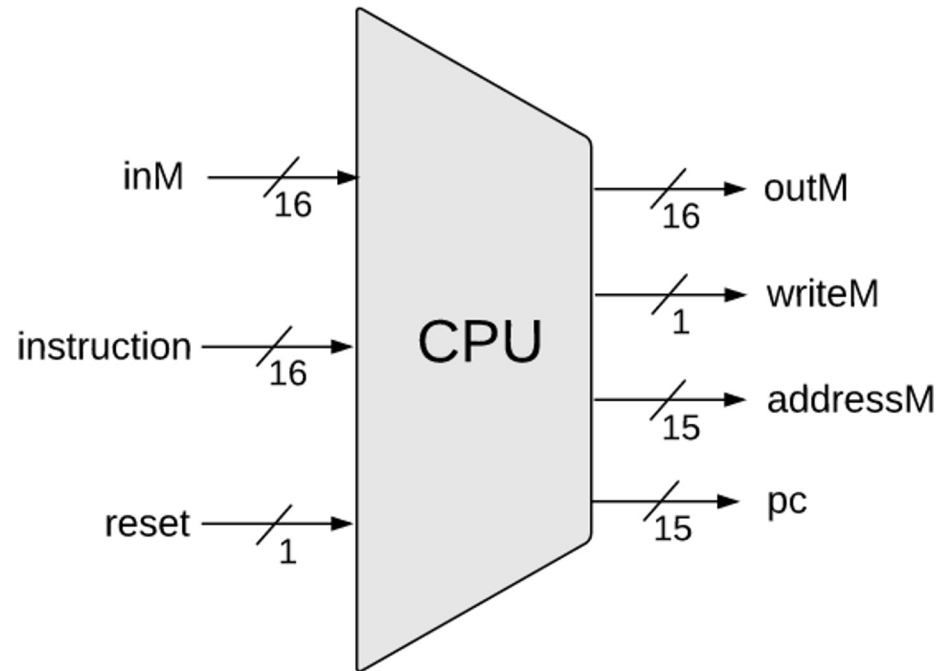
- Midterm Topics Brainstorming

❖ Project 6 Overview

- Project Tips and Workflow

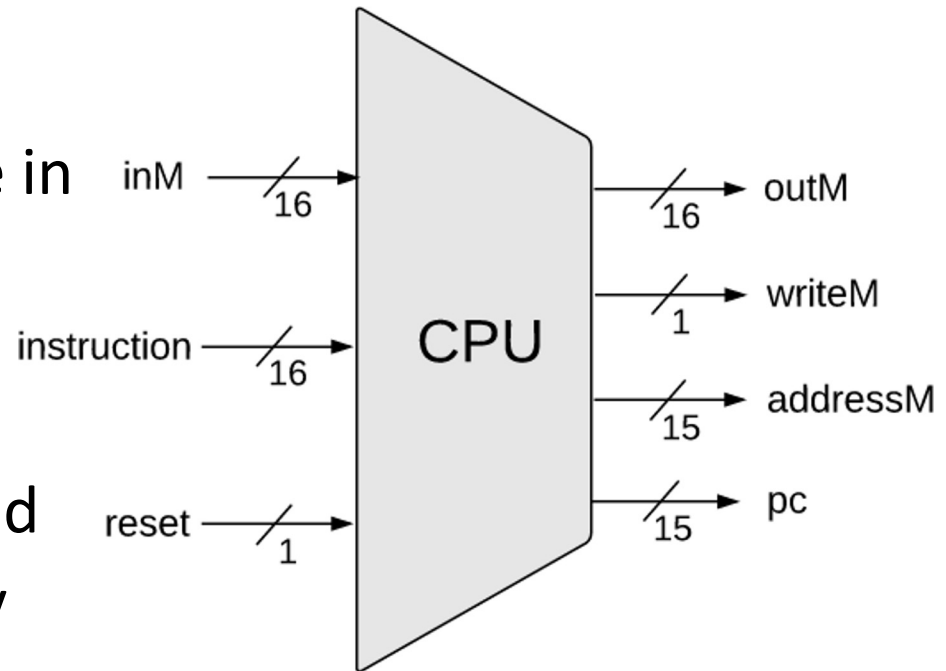
Hack CPU Interface Inputs

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

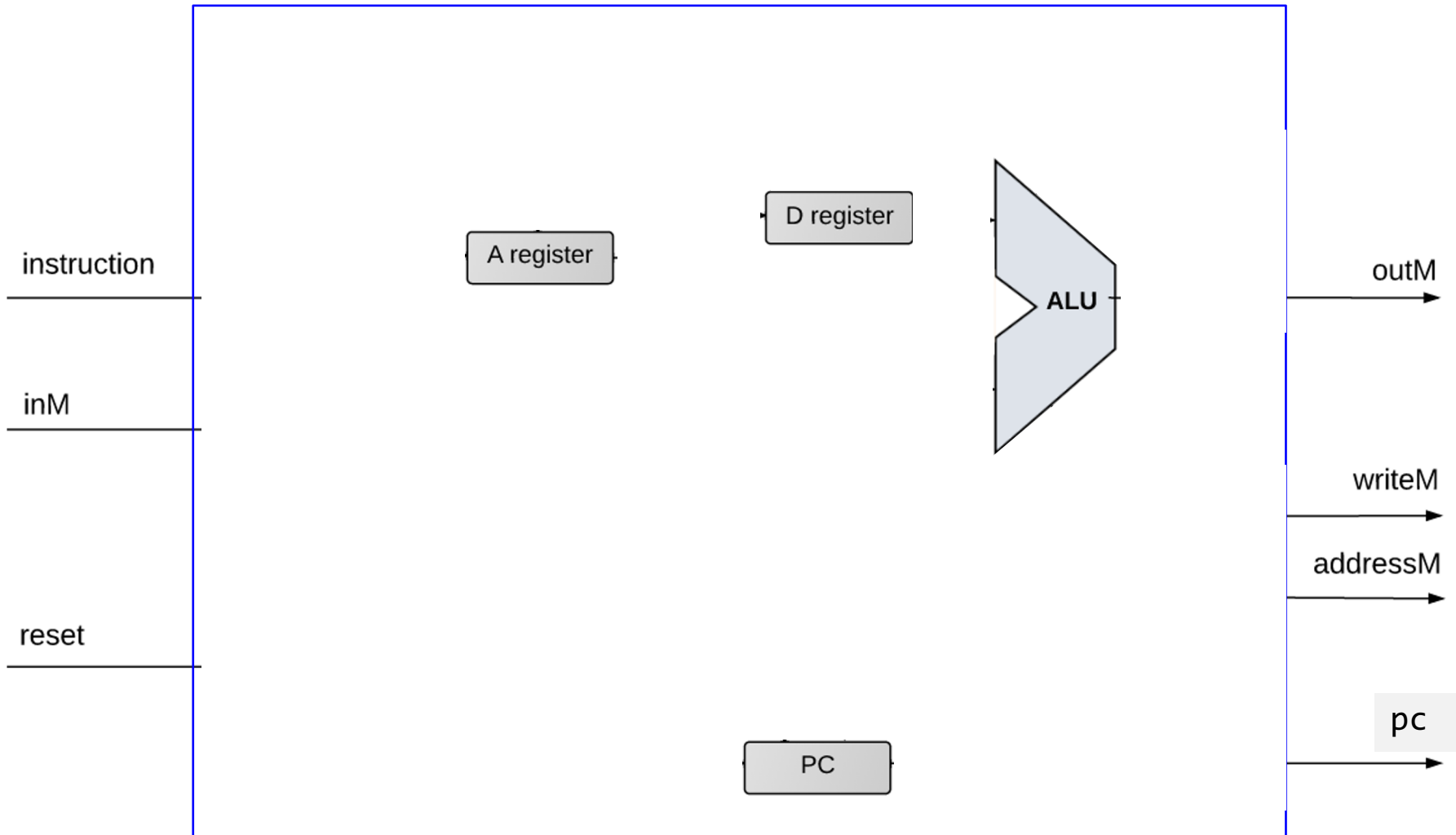


Hack CPU Interface Outputs

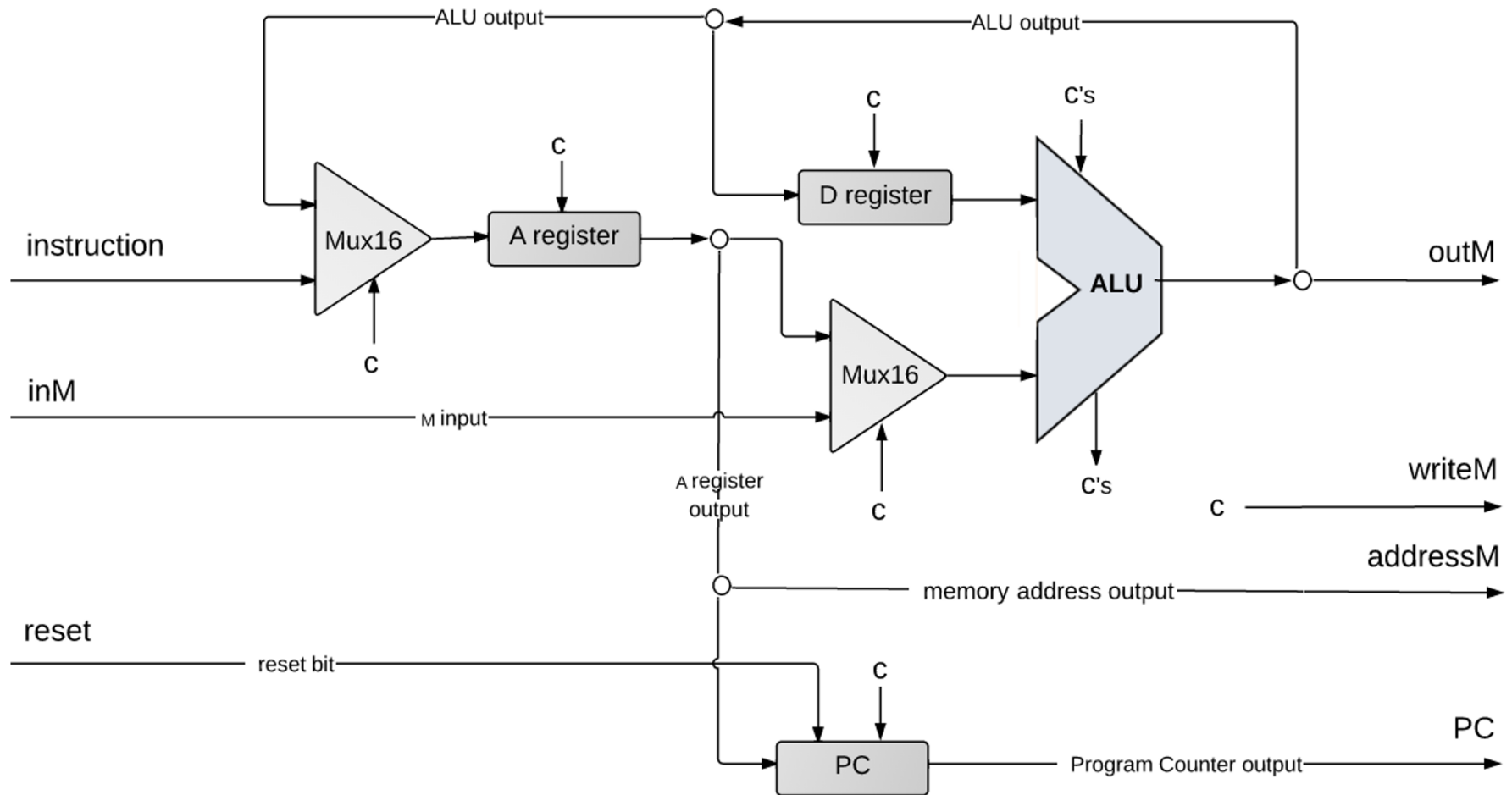
- ❖ **outM**: value used to update memory if writeM is 1
- ❖ **writeM**: if 1, update value in memory at addressM with outM
- ❖ **addressM**: address to read from or write to in memory
- ❖ **pc**: address of next instruction to be fetched from memory



Hack CPU Implementation

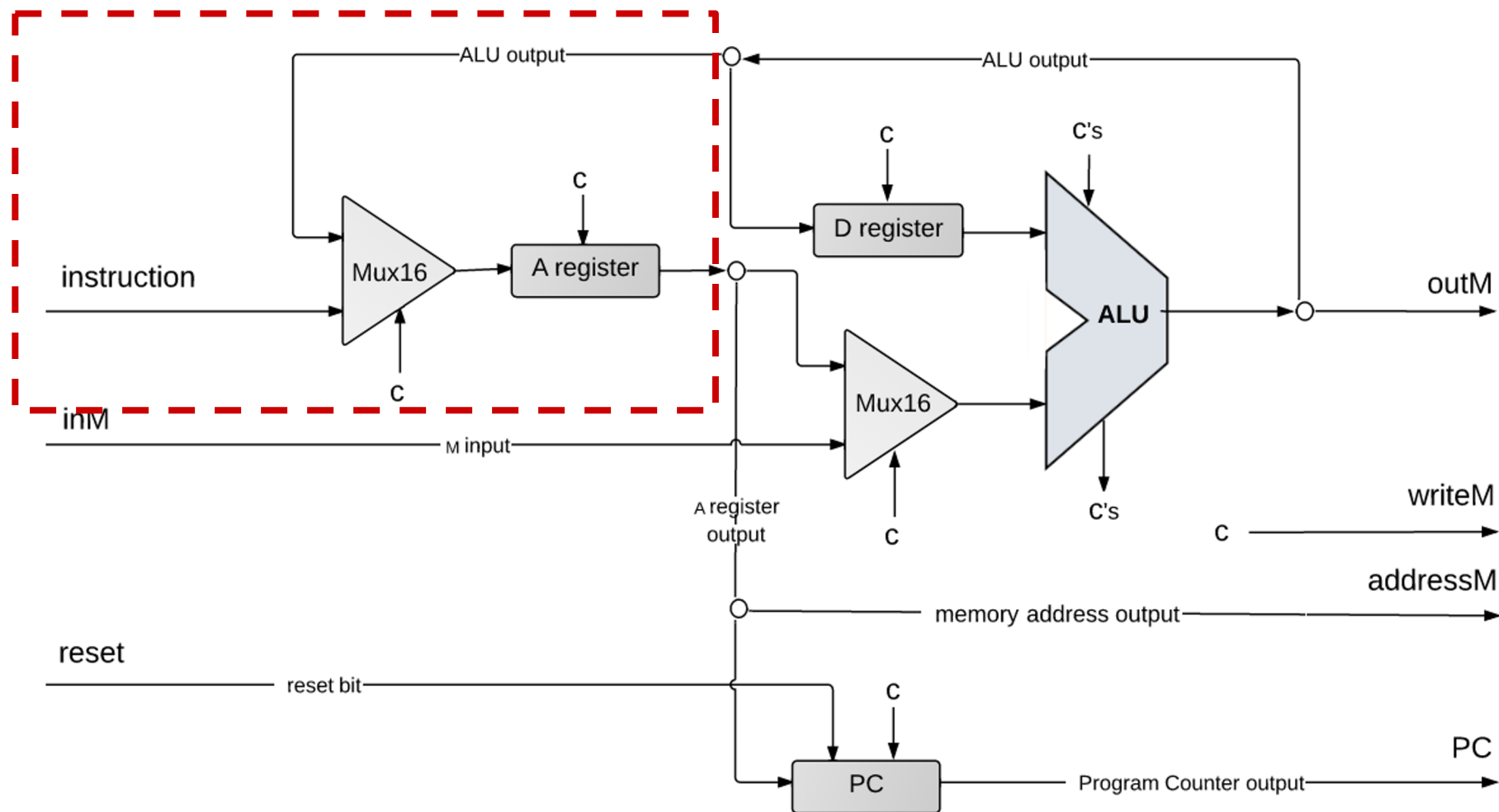


Hack CPU Implementation



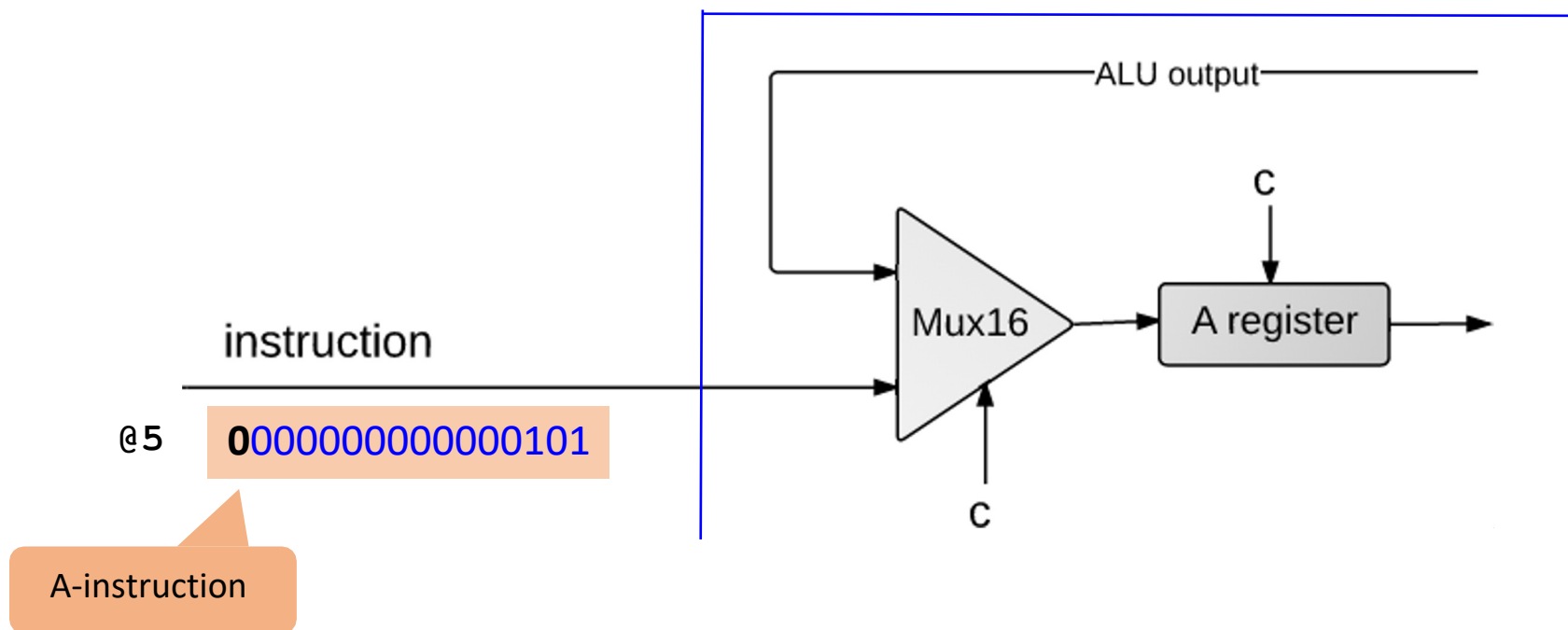
(each "c" symbol represents a control bit)

CPU Operation: Instruction Handling

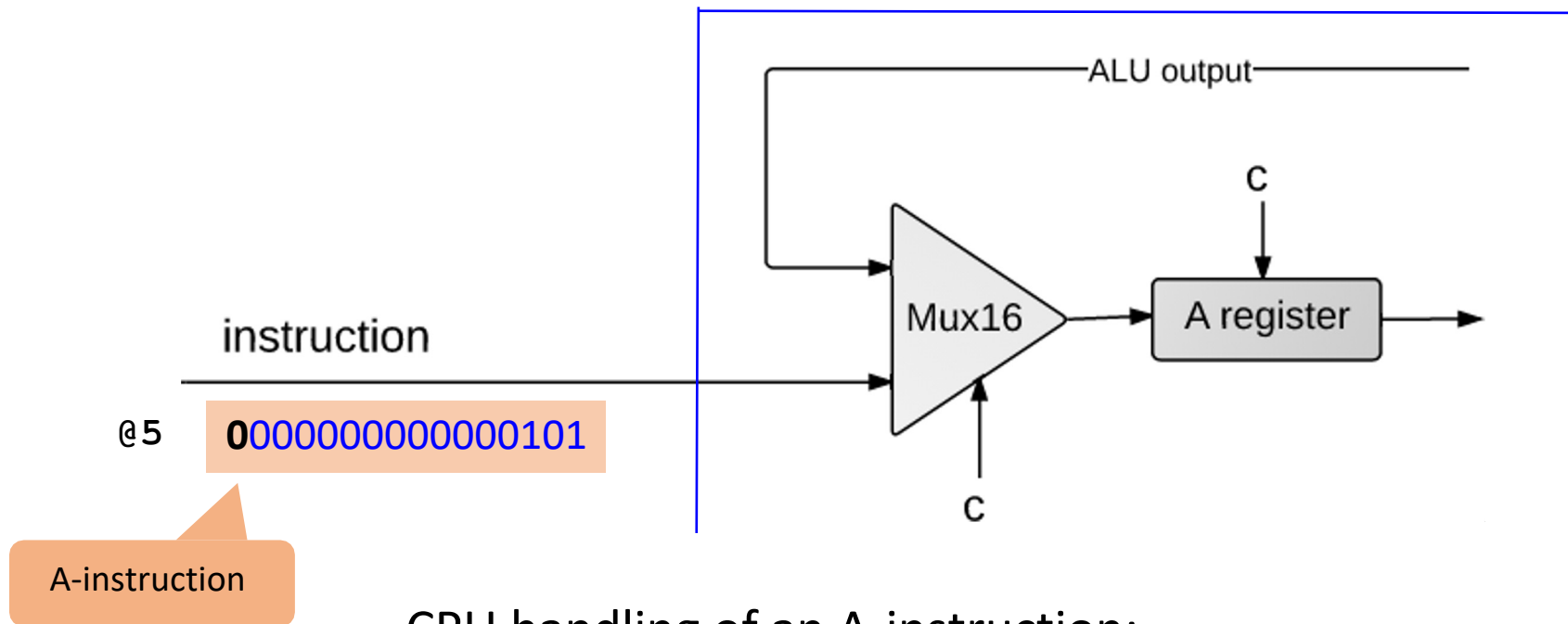


(each "c" symbol represents a control bit)

CPU Operation: Instruction Handling



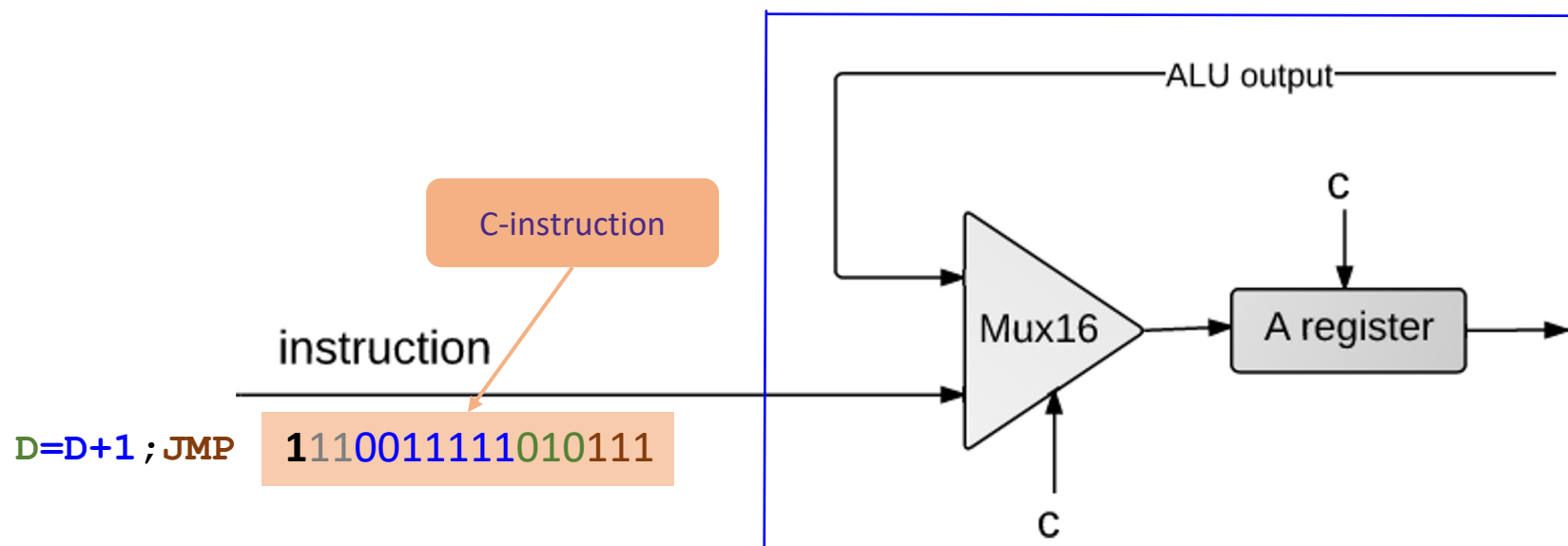
CPU Operation: Instruction Handling



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`

Family:
C-Instruction

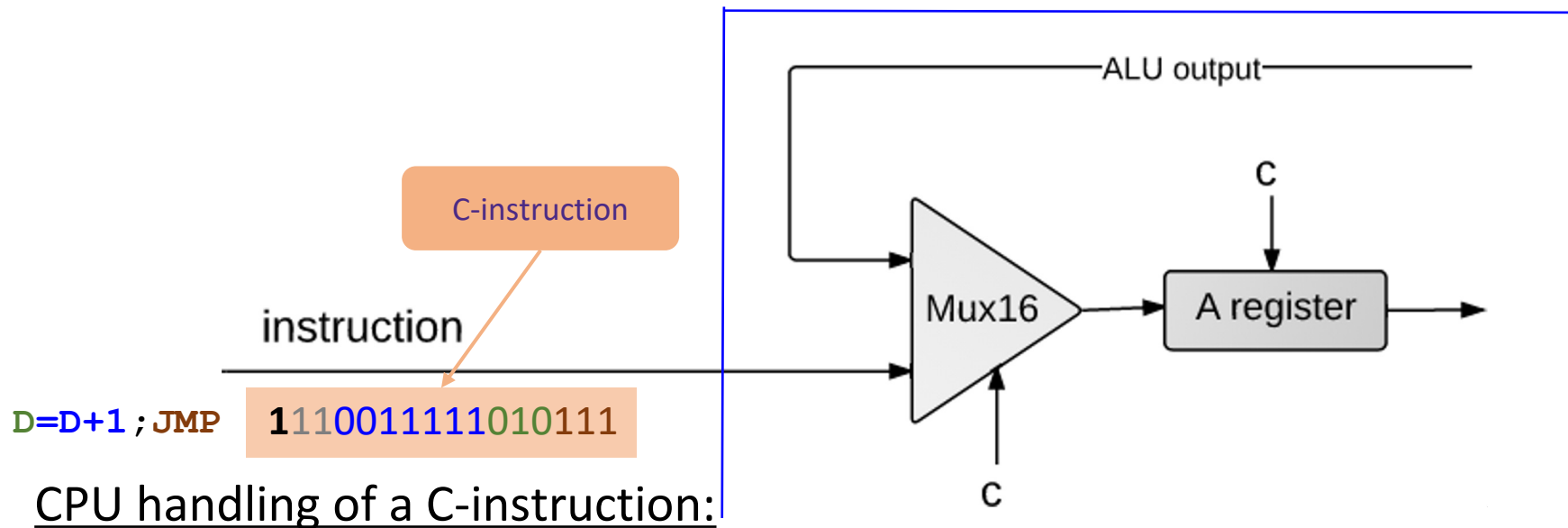
Unused

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

Dest:
Where to store
result

Jump:
Condition for
jumping

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

(when a=0) <i>comp mnemonic</i>	c1	c2	c3	c4	c5	c6	(when a=1) <i>comp mnemonic</i>
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	M+1
A+1	1	1	0	1	1	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

Comp:

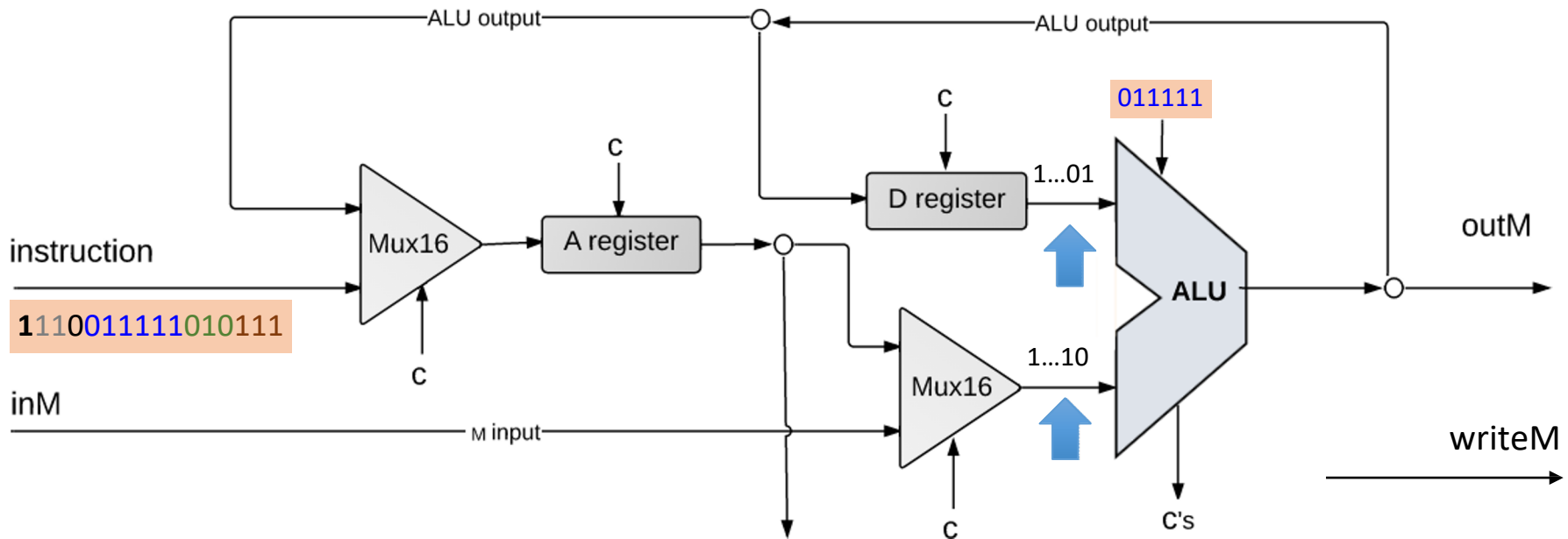
ALU Operation (a bit chooses between A and M)

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`

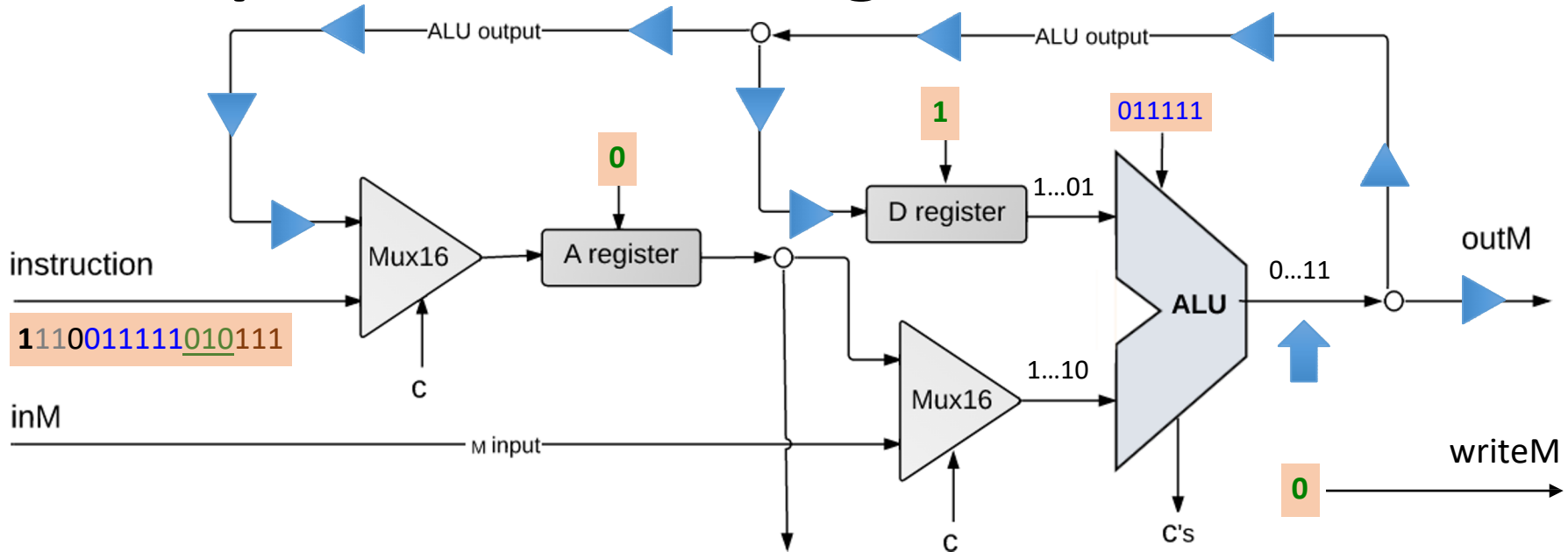
❖ Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

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

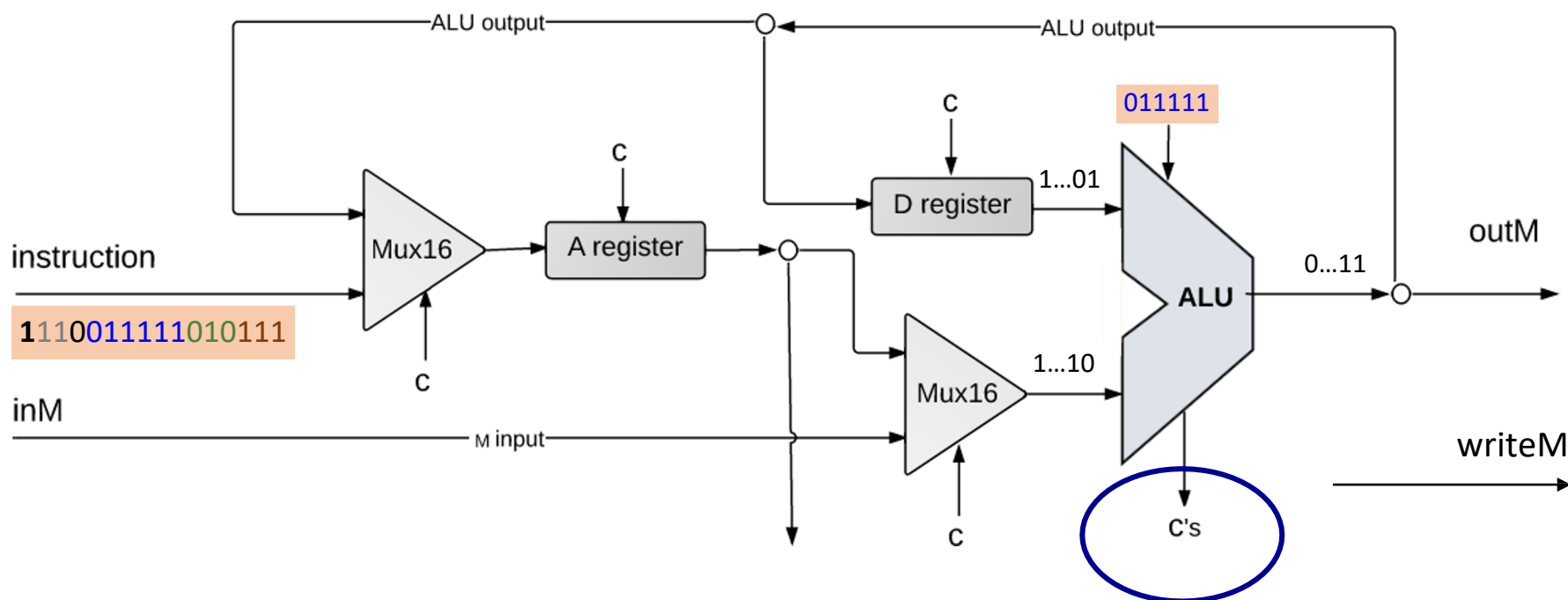
CPU Operation: Handling C-Instructions



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**

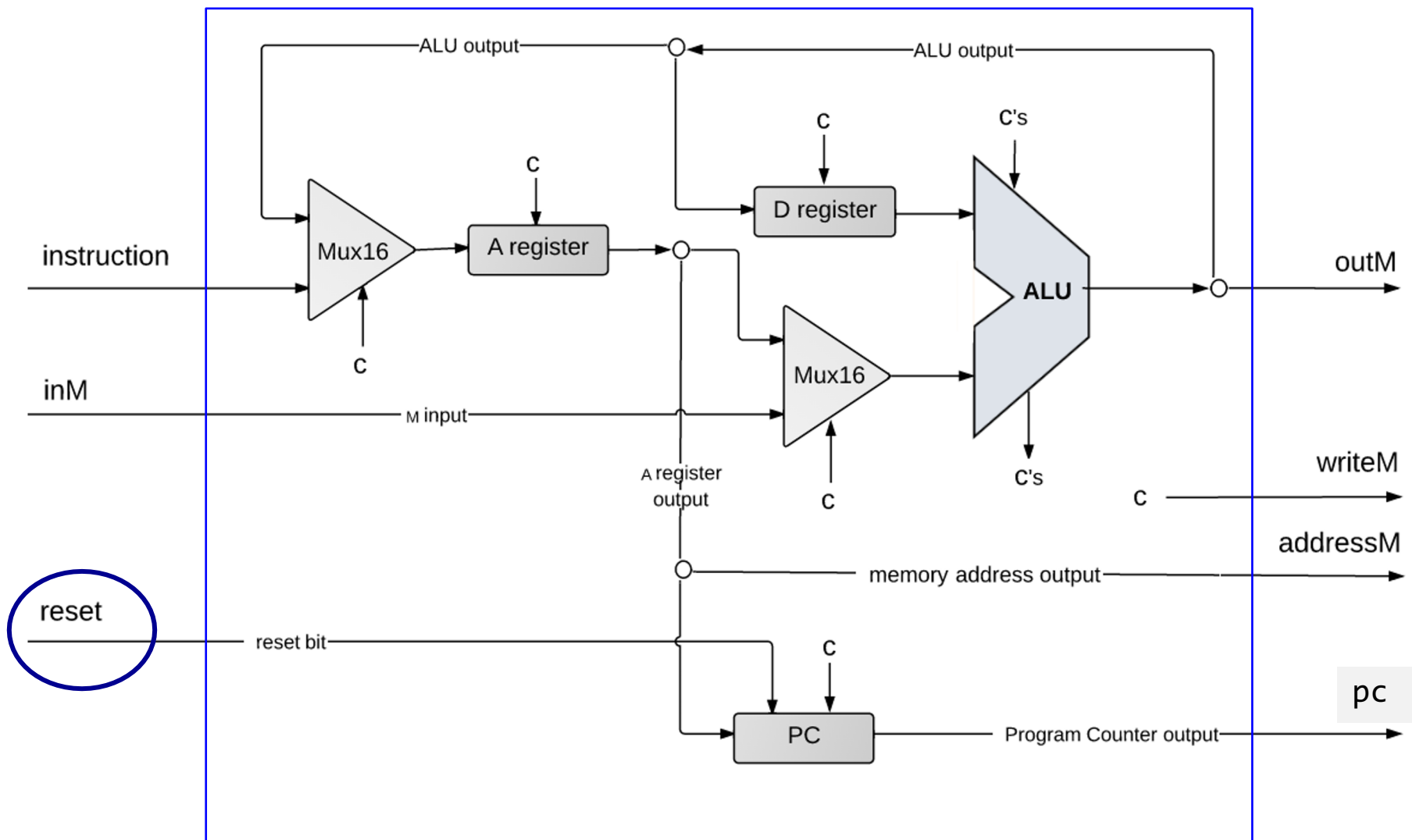
CPU Operation: Handling C-Instructions



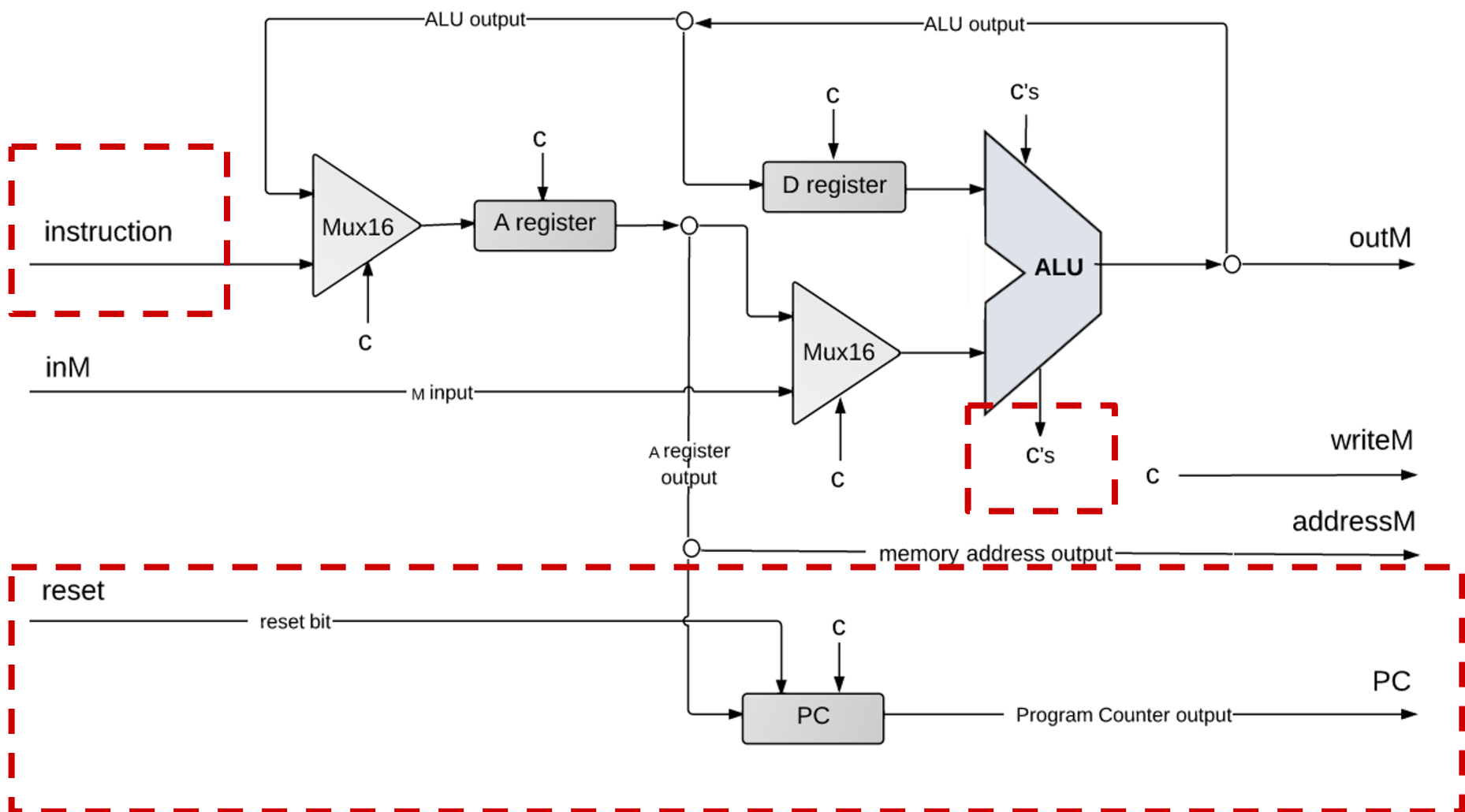
ALU control outputs:

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

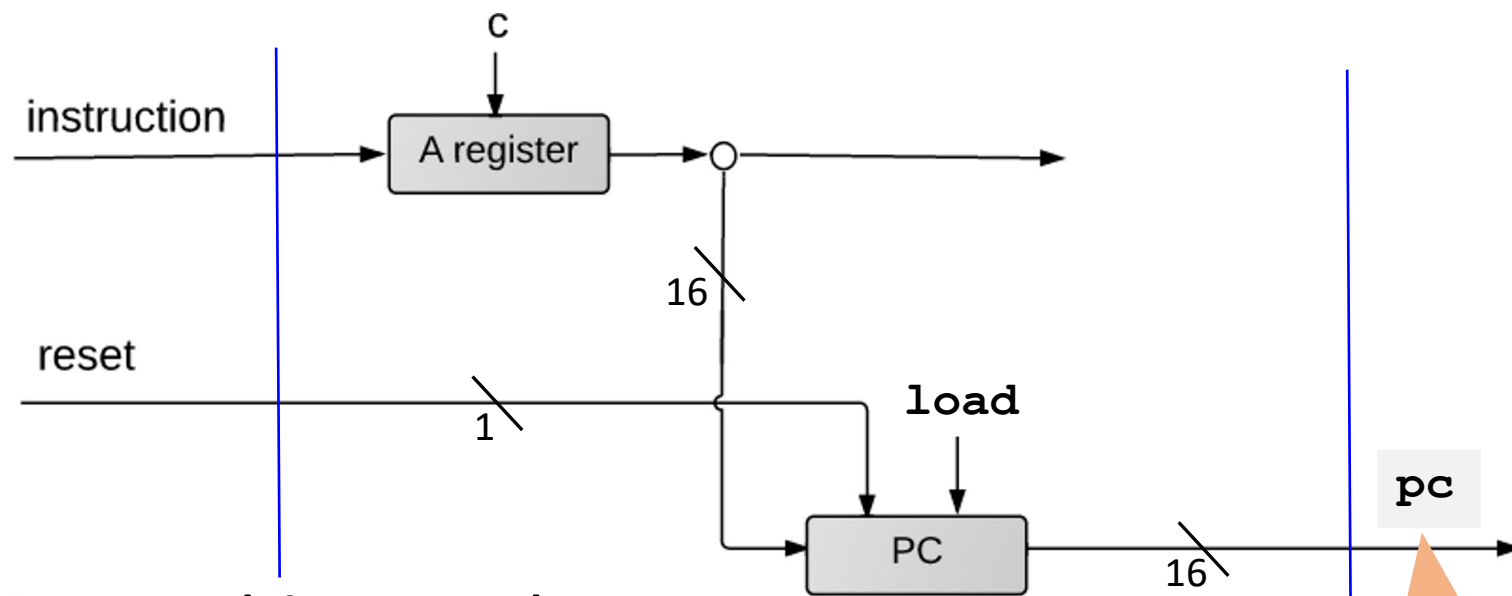
CPU Operation: Control



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`

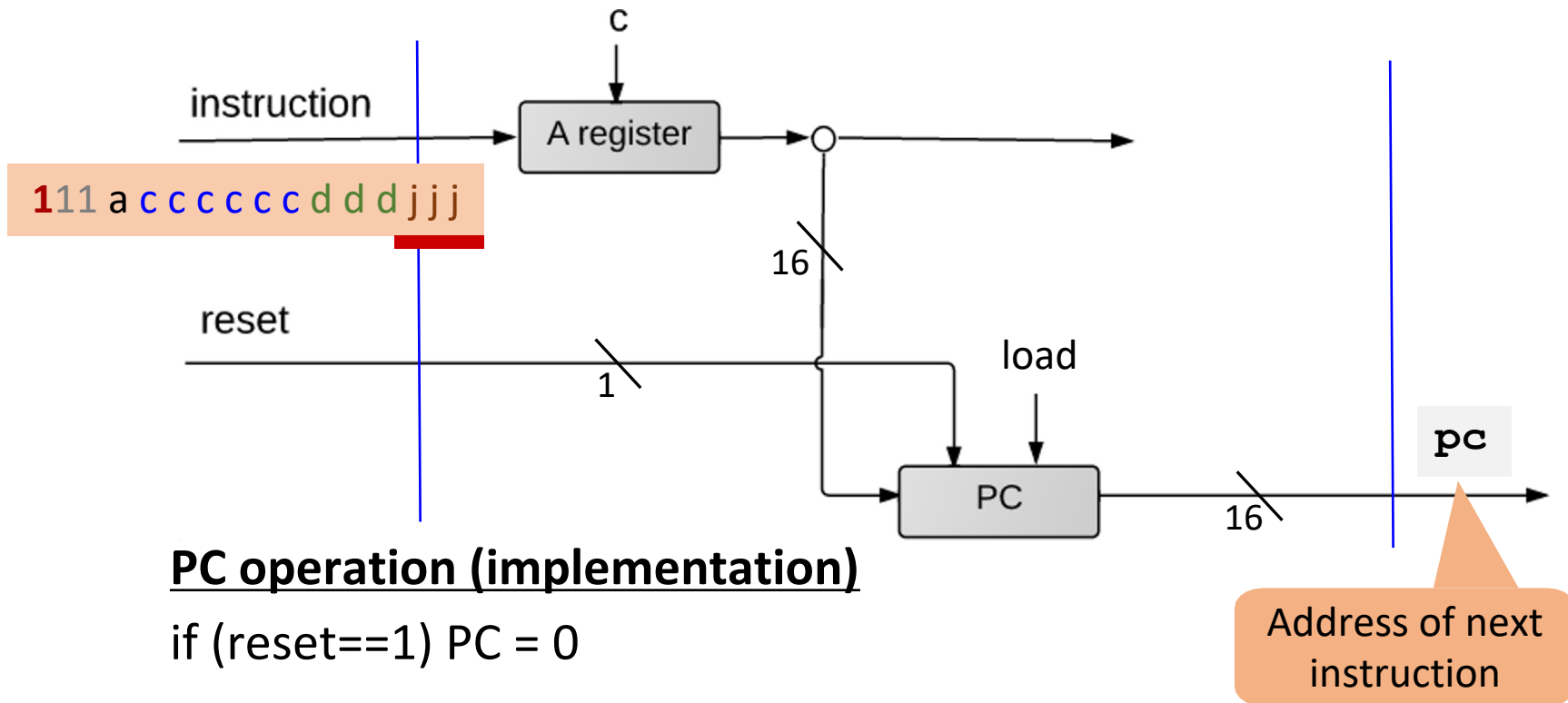
❖ Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

Jump:
Condition for
jumping

Chapter 4

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

CPU Operation: Control



PC operation (implementation)

if (reset==1) PC = 0

else

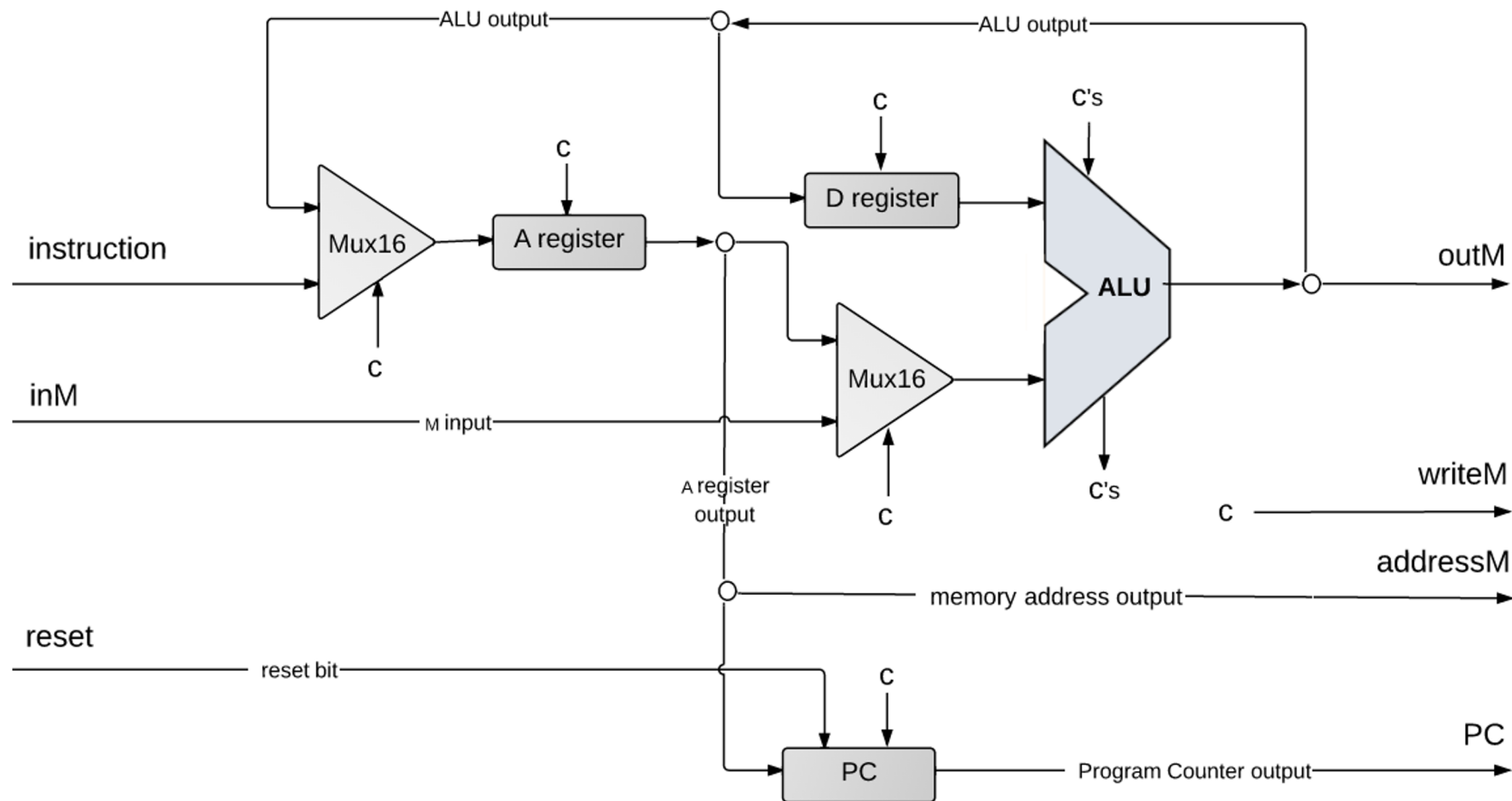
// In the course of handling the current instruction:

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 so far in class, what are topics, concepts, questions that you might expect to show up on next week's CSE 390B midterm?

Previous CSE 390B Midterms

- ❖ Several midterms from previous quarters
 - 20sp midterm likely more difficult than midterm this quarter
 - Rest of the midterms are more like what this quarter's midterm will look like

- ❖ 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

Lecture 10 Reminders

- ❖ **Project 5 due tonight (2/2) at 11:59pm**
- ❖ **CSE 390B midterm next Friday (2/9) during lecture**
- ❖ Project 6 (Mock Exam Problem & Building a Computer) released today, due in two Fridays (2/16) at 11:59pm
- ❖ Eric has office hours after class in CSE2 153
 - Feel free to post your questions on the Ed board as well