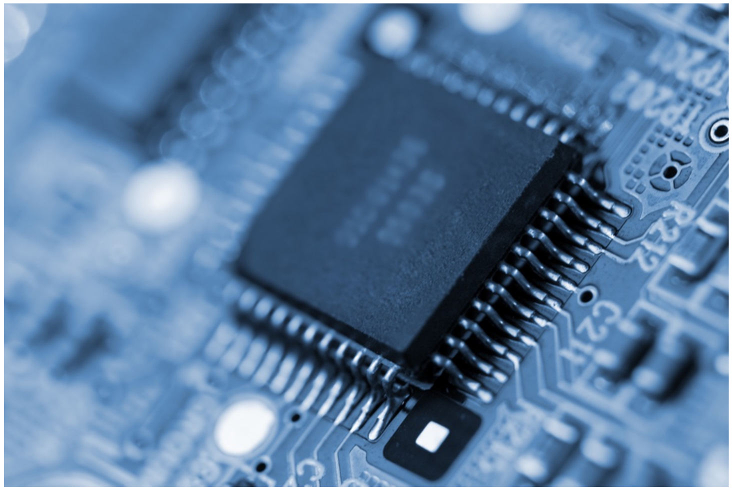
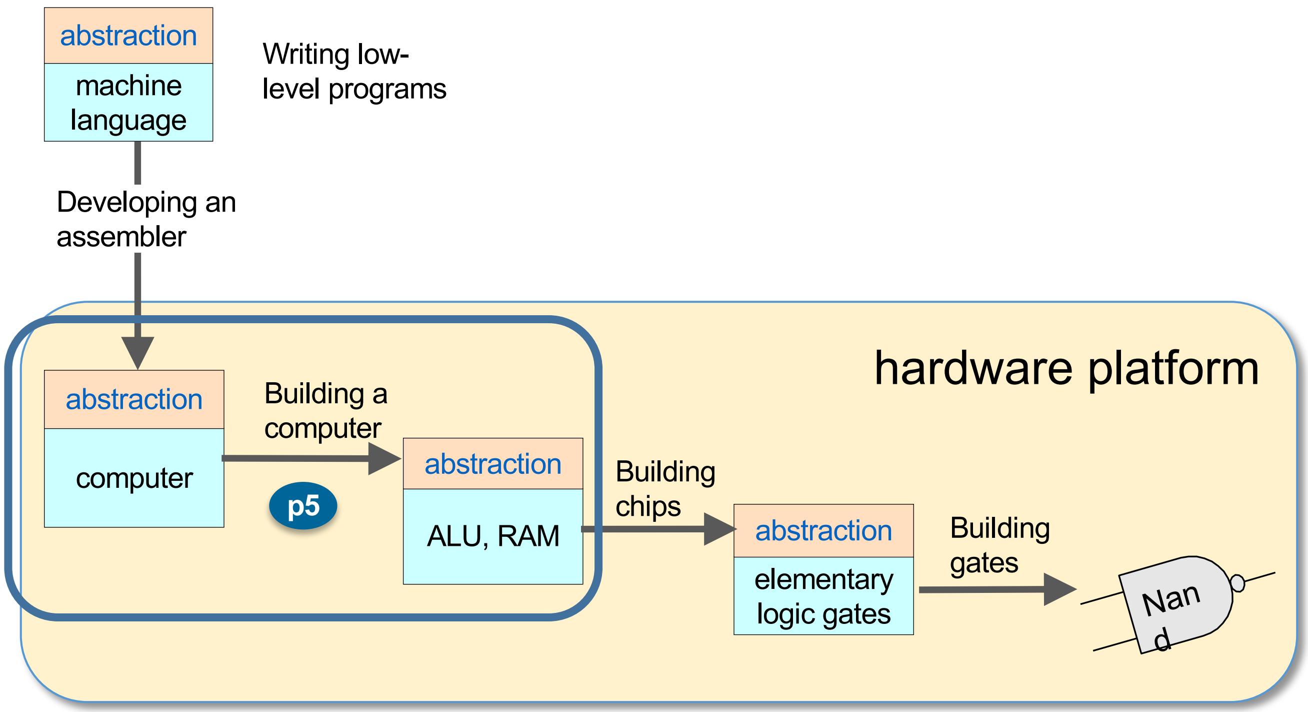
Computer Processors

Computer Architecture



Lecture 12


abstraction
machine language
Writing low-level programs
Developing an assembler
hardware platform
abstraction
computer
Building a computer
p5
abstraction
ALU, RAM
Building chips
abstraction
elementary logic gates
Building gates
Nand

Modern computers (20th century)

UNIVERSITY OF LEEDS


John Von Neumann

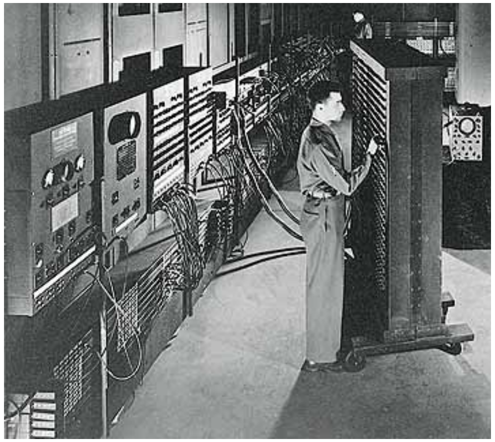

John Mauchly


Presper Eckert


John Atanassof

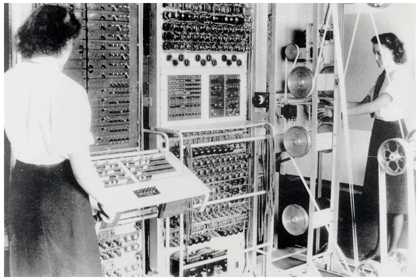

Howard Aiken


Konrad Zuse



ENIAC: First digital, programmable, stored program computer

University of Pennsylvania, 1946,

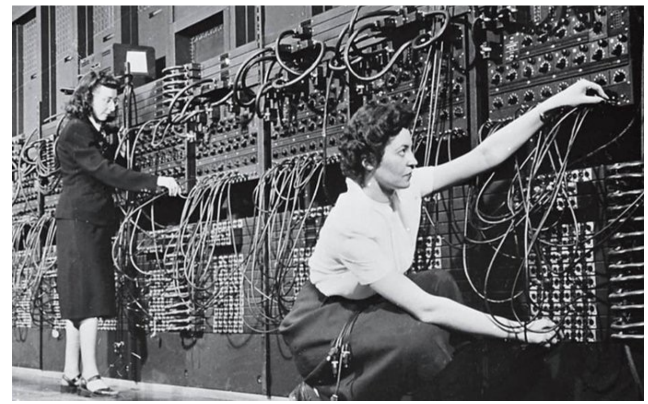



Tommy Flowers

Colossus: First digital, programmable, computer, UK, 1945

Modern computers (20th century)

Kathleen McNulty, Jean Jennings, Frances Snyder, Marlyn Wescoff, Frances Bilas, Ruth Lichterman



ENIAC Women

Pioneered reusable code, subroutines, flowcharts, and many other programming innovations

Compilation pioneers

Grace Hopper

Adele Koss

Stored program computer

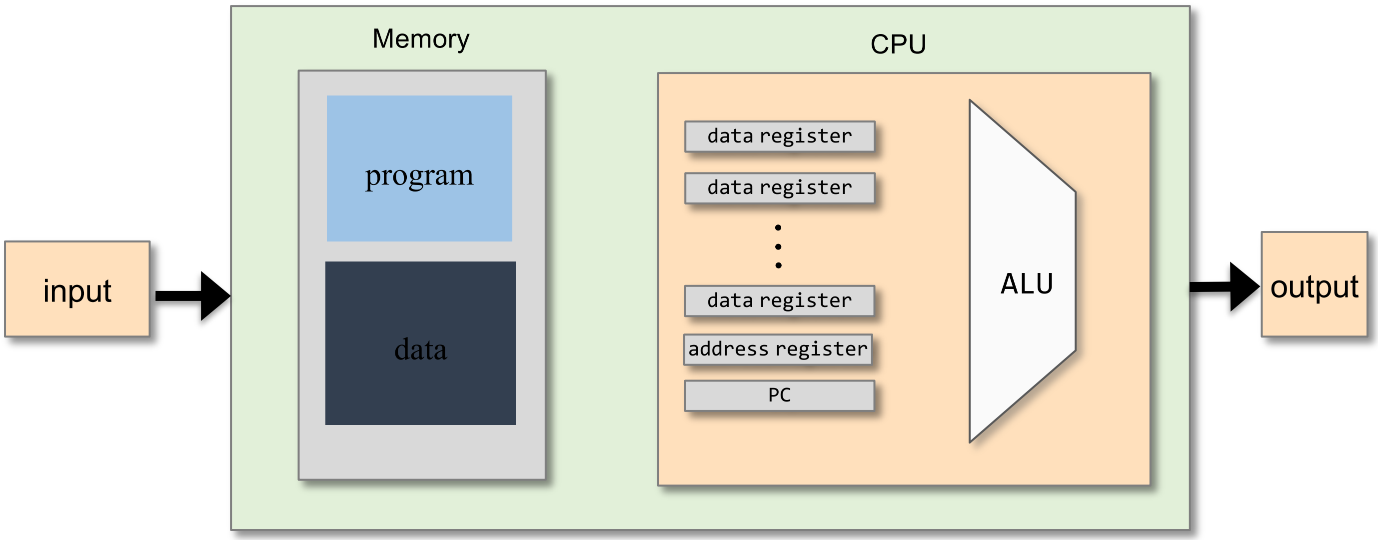
- Computer has a fixed set of instructions
- Instructions can be used and combined constructing arbitrarily complex programs
  - Games
  - Scientific calculations
  - Communication
- The logic of the program is not embedded in the hardware it is stored in program memory
- The computer can be programmed and reprogrammed to complete a task

Von Neumann architecture

UNIVERSITY OF LEEDS

- Named after John von Neumann (1945)
- A description of an abstract machine containing
  - A CPU (including ALU and registers)
  - Control unit
  - Program counter
  - Memory (data and instruction)
  - Input/Output devices
- Any computer where the fetch-instruction and data operations can't occur concurrently as they share a common bus

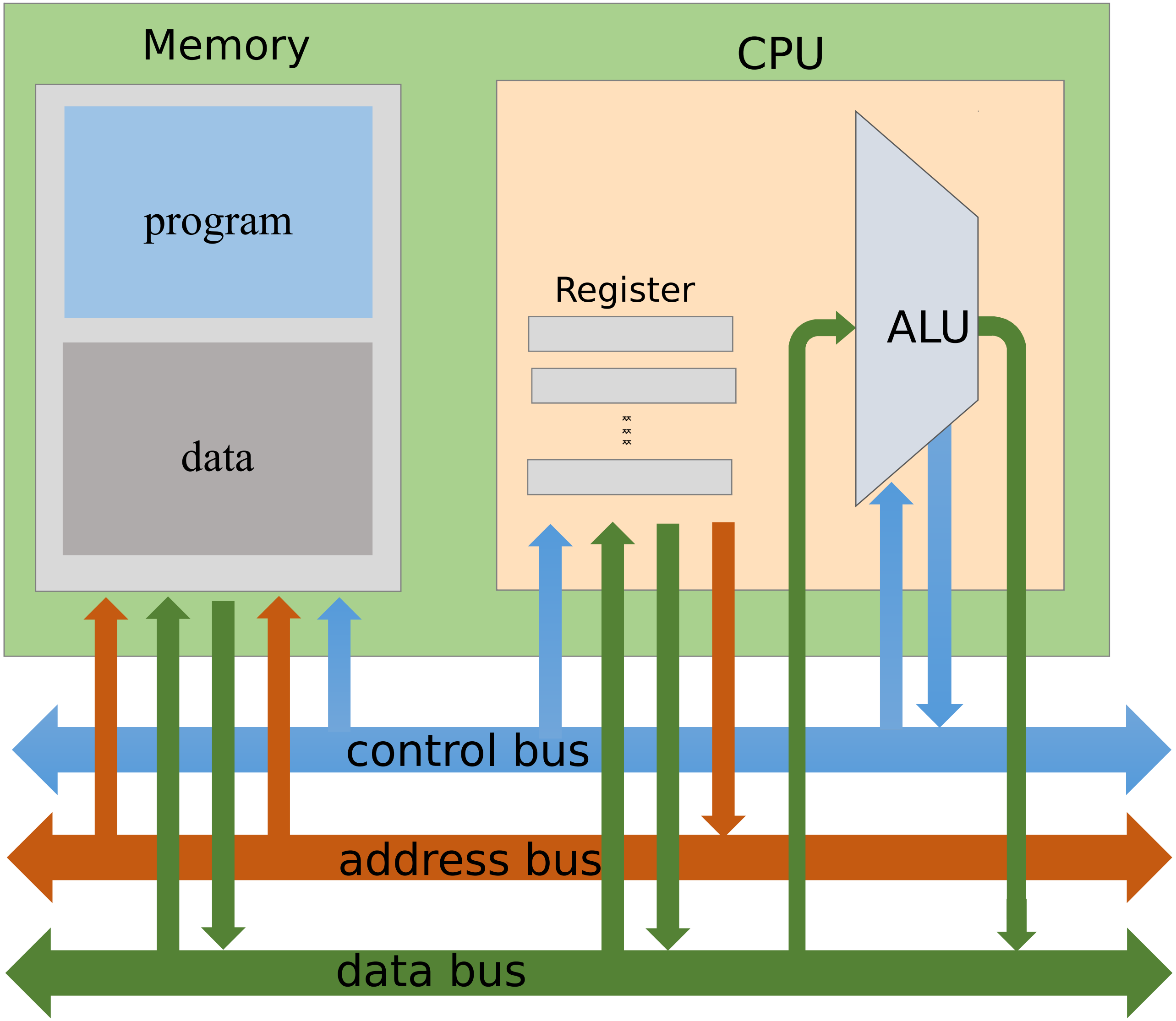
Typical computer architecture



- Stored program concept
- General-purpose

Computer Architecture

UNIVERSITY OF LEEDS



Three types of information that's usually passed around the system:

1. Data

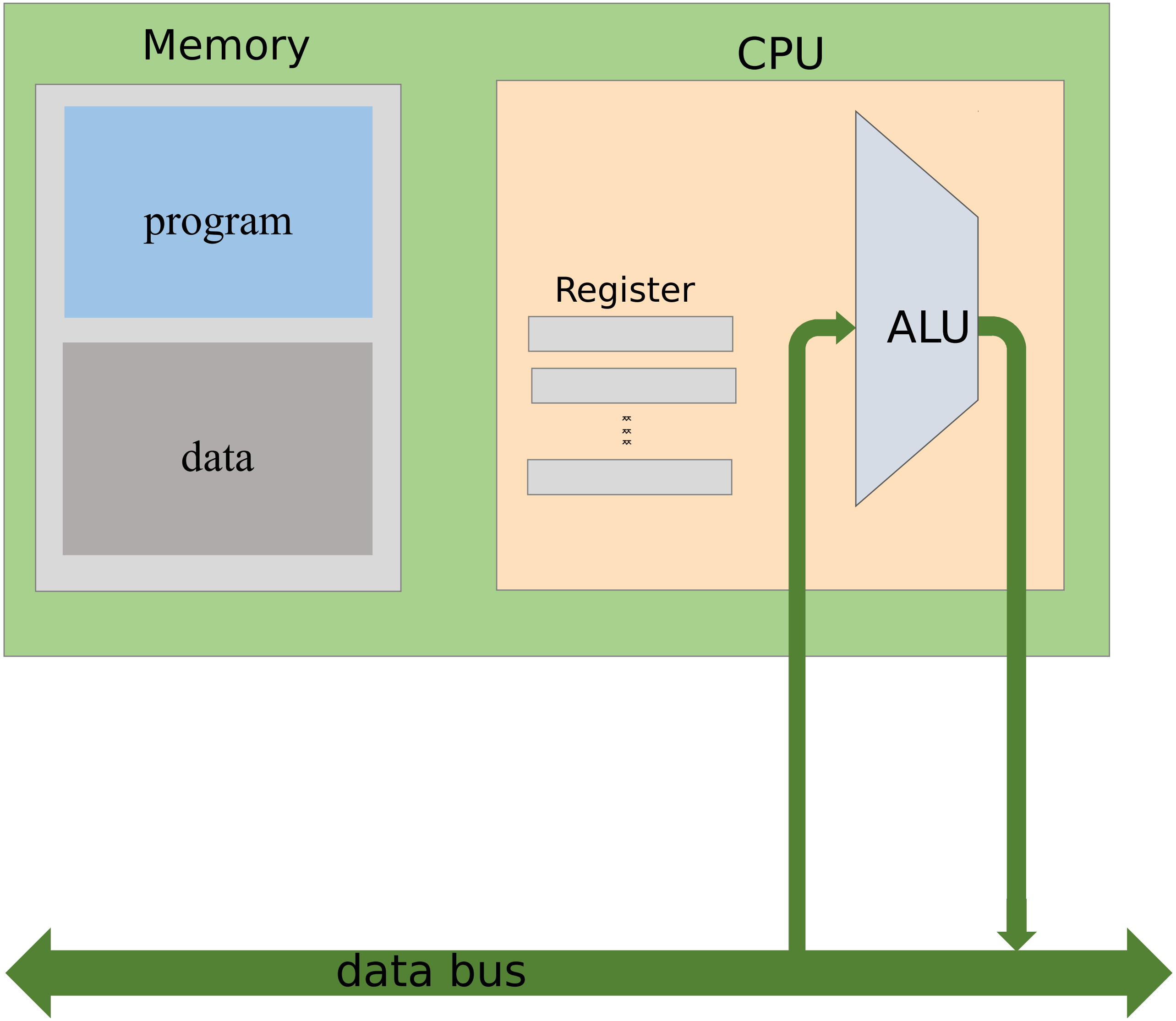
2. Addresses

3. Control

Each one of these pieces of information is usually implemented by wires, called a bus.

Computer Architecture (ALU and Data Bus)

UNIVERSITY OF LEEDS

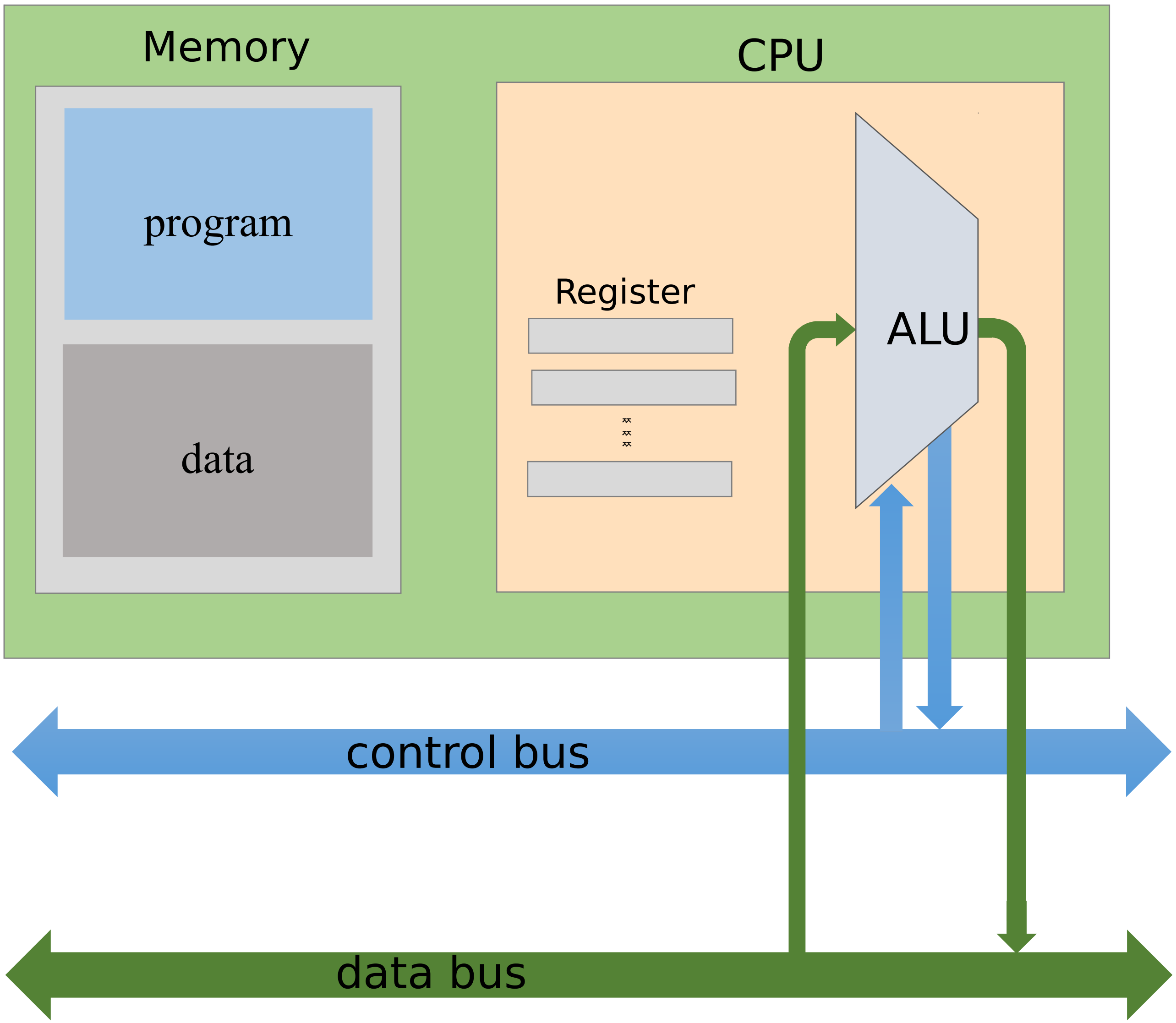


ALU-Functions: accept numbers to e.g. add them, subtract them, do some logical operations on them …

Thus, we need to have some information from the databus connect into the ALU.

Then, ALU feeds the output value back into the databus, where information goes to other places that are also connected to the databus, like the memory or the registers.

Computer Architecture (ALU and Control Bus)

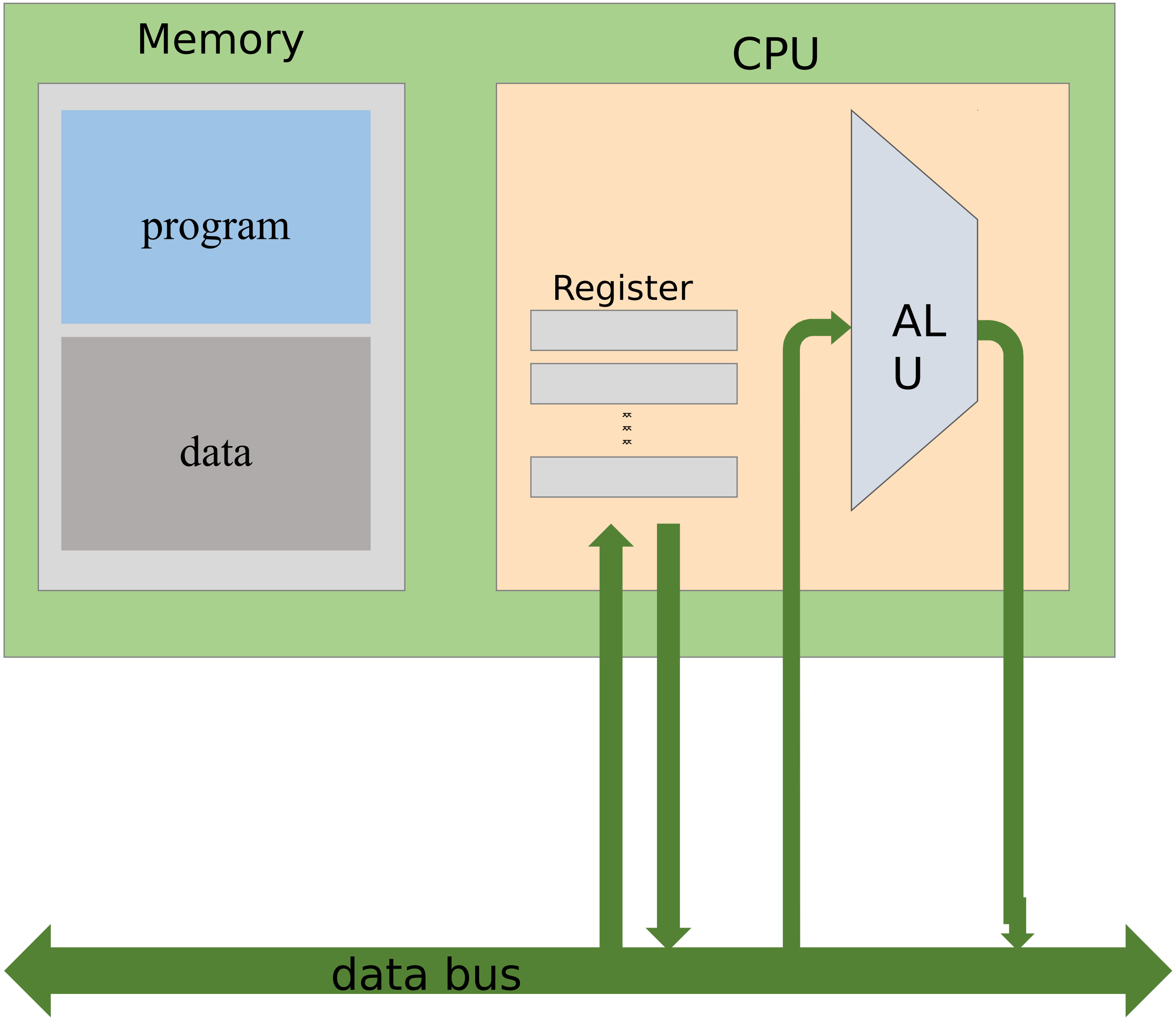


ALU has to get information from the control bus specifying the type of operation that it has to do.

According to the results of the arithmetic or logical operations that ALU does, it has to be able to tell the other parts of the system what to do. Thus, we have to take some information from the ALU and feed it back to control the rest of the system.

Computer Architecture (Reg. and Data Bus)

UNIVERSITY OF LEEDS



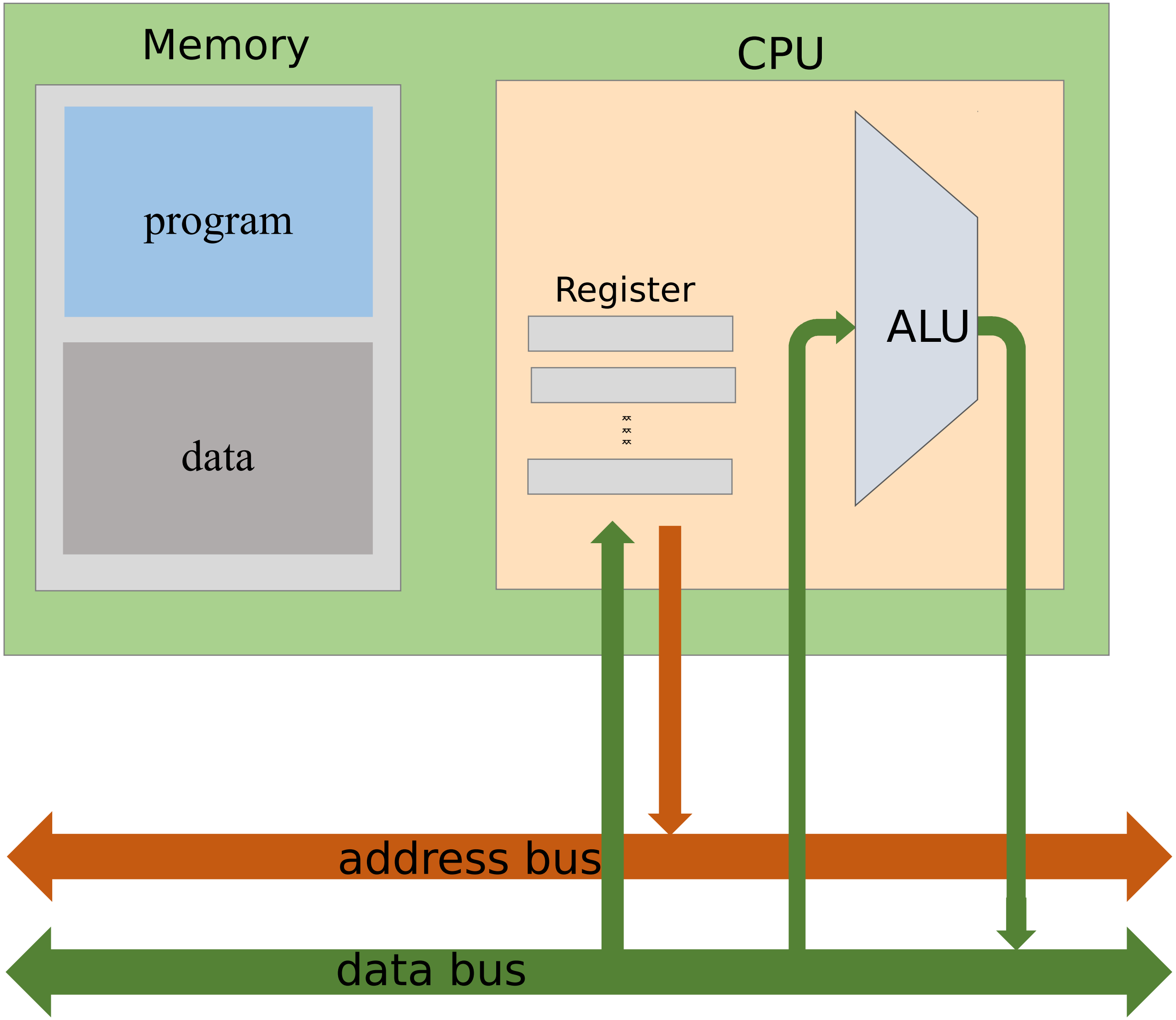
Register “store” intermediate results

→ We must be able to put date from the data bus into the registers and then data from the register must be fed back into the data bus

(Note, all registers must be connected to the data bus).

Computer Architecture (Reg. and Address Bus)

UNIVERSITY OF LEEDS



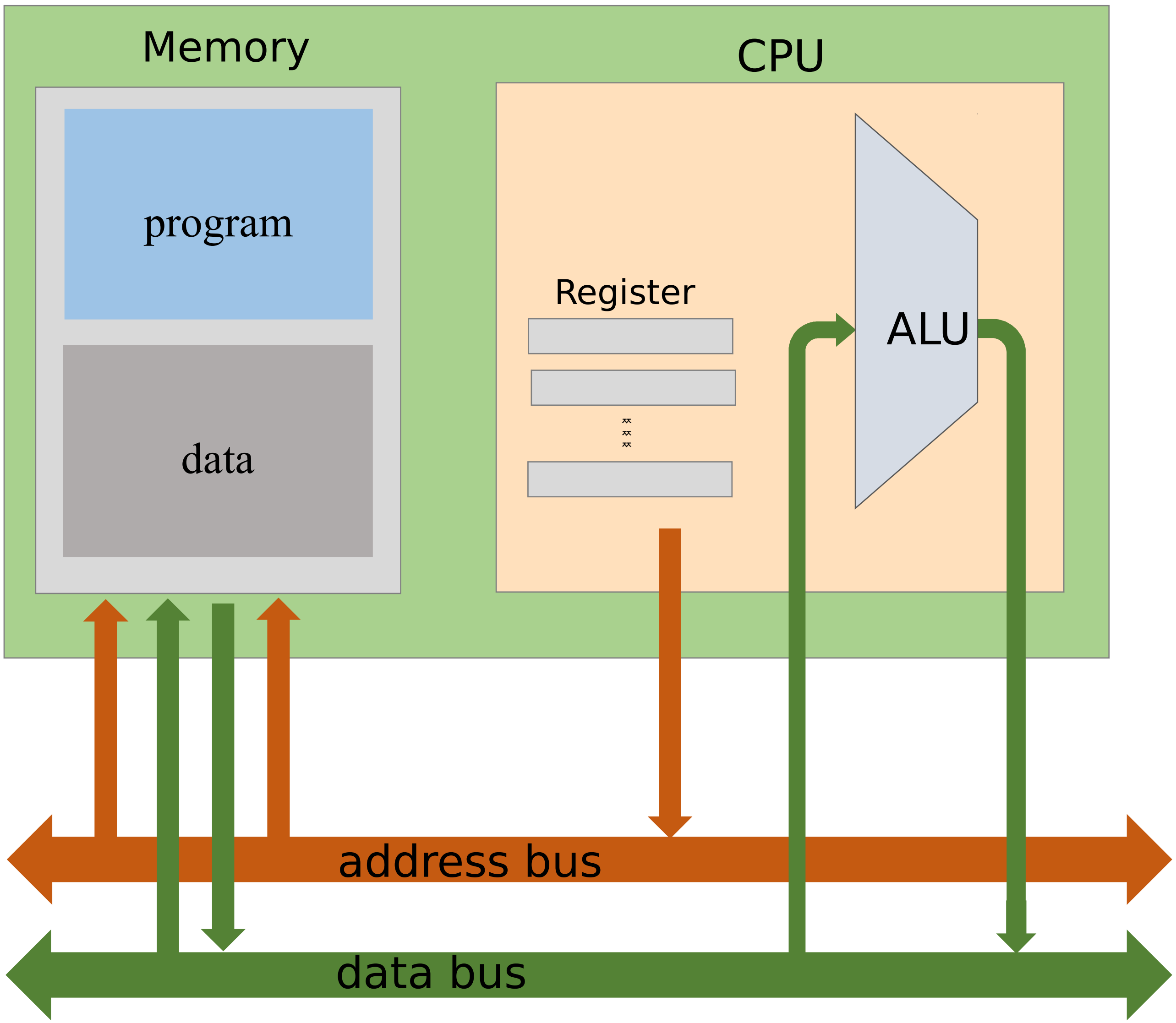
Register "store" addresses

We actually achieve indirect addressing into a RAM or jump into a ROM address by putting numbers (addresses) into a register that specifies where we want to access.

→ registers are connected to the address bus (which controls again e.g. the memories)

Computer Architecture
(Memory and Address Bus / Data Bus)

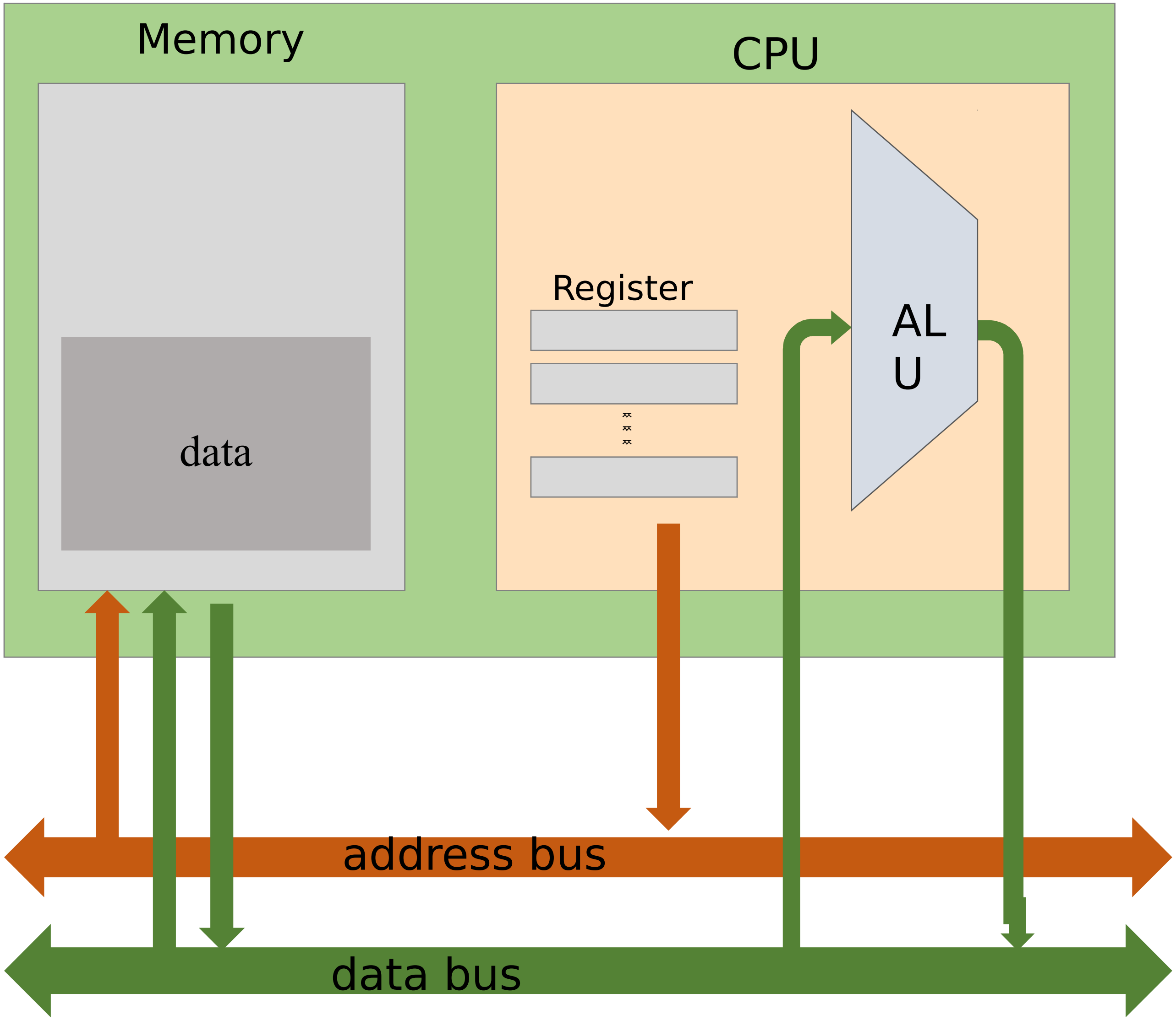
UNIVERSITY OF LEEDS



We always need to specify what address of the memory are we going to be working with.
→ specified by the *address bus*

Once we actually work with a certain address, we need to be able to read it or write into it (get information from it or put information into it)
→ connected to the *data bus.*

Computer Architecture
(DATA Memory and Address Bus / Data Bus)



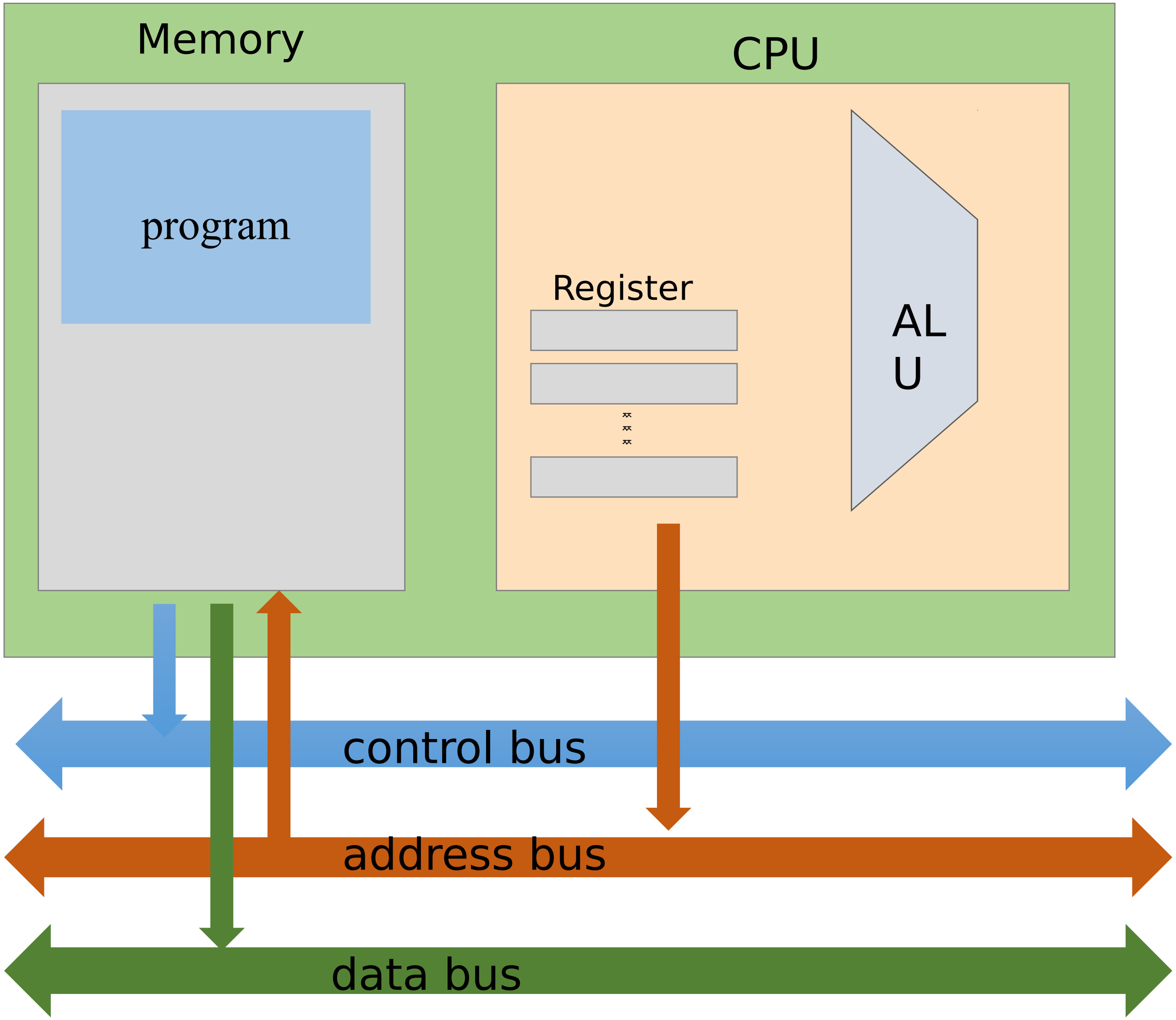
Data Memory:

* it's going to get an address of a data piece that needs to be operated upon
→ *address bus*

* we need to write and read into it → *data bus*.

Computer Architecture
(PRG Memory and Address-, Data-, Control Bus)

UNIVERSITY OF LEEDS



Program Memory:

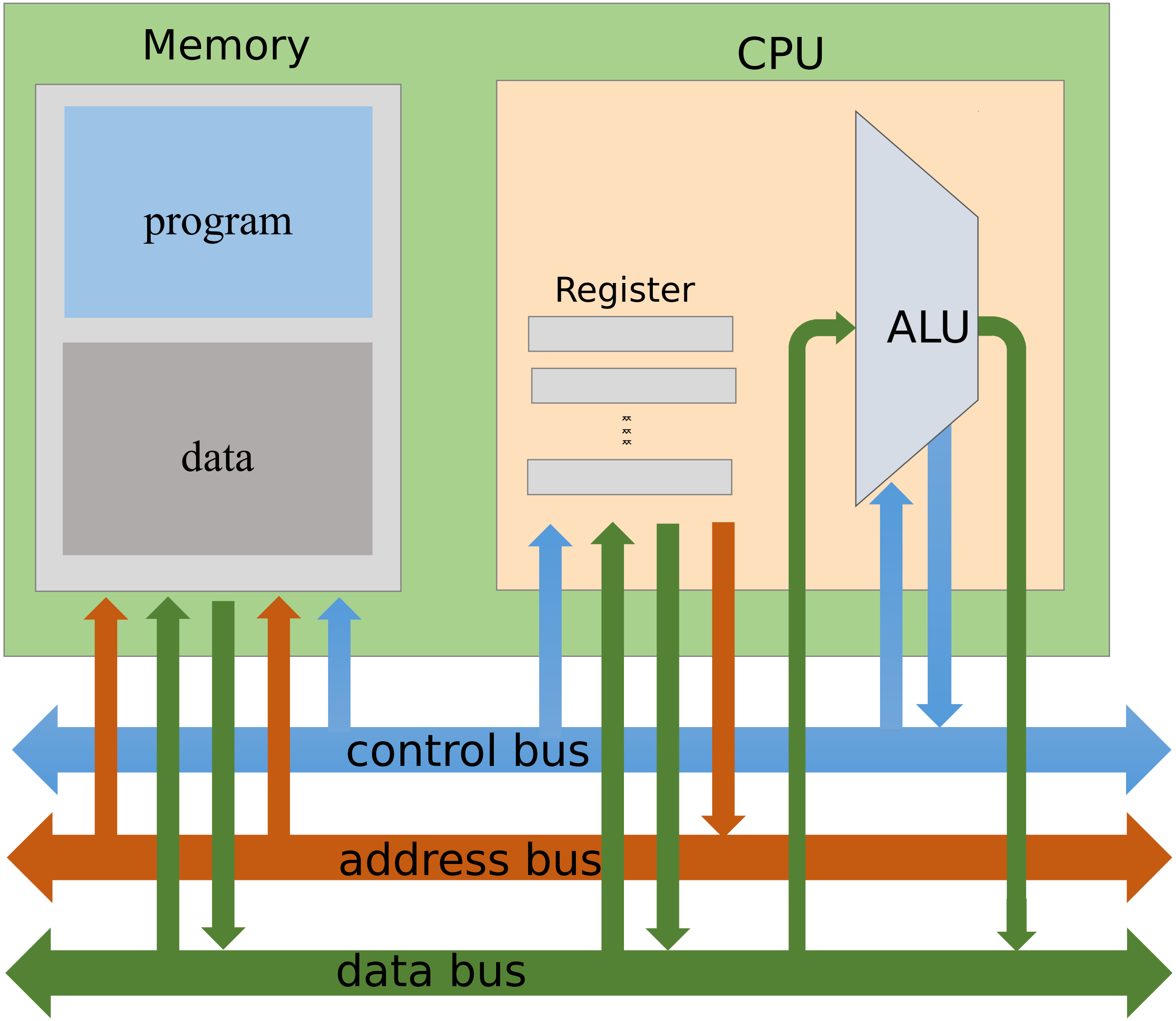
* We need to put the address of the next program instruction into the program memory and then get the instructions from there
→ *address bus*

* Now the instructions that we get from the program memory, both may have data in it (e.g. numbers that we need to add, and so on)
→ data *bus*

* the program instruction tells the rest of the system what to do. So we need to be able feed this information into the control bus
→ *control bus*

Computer Architecture

UNIVERSITY OF LEEDS



Three types of information that's usually passed around the system via

1. Data Bus

2. Address Bus

3. Control Bus

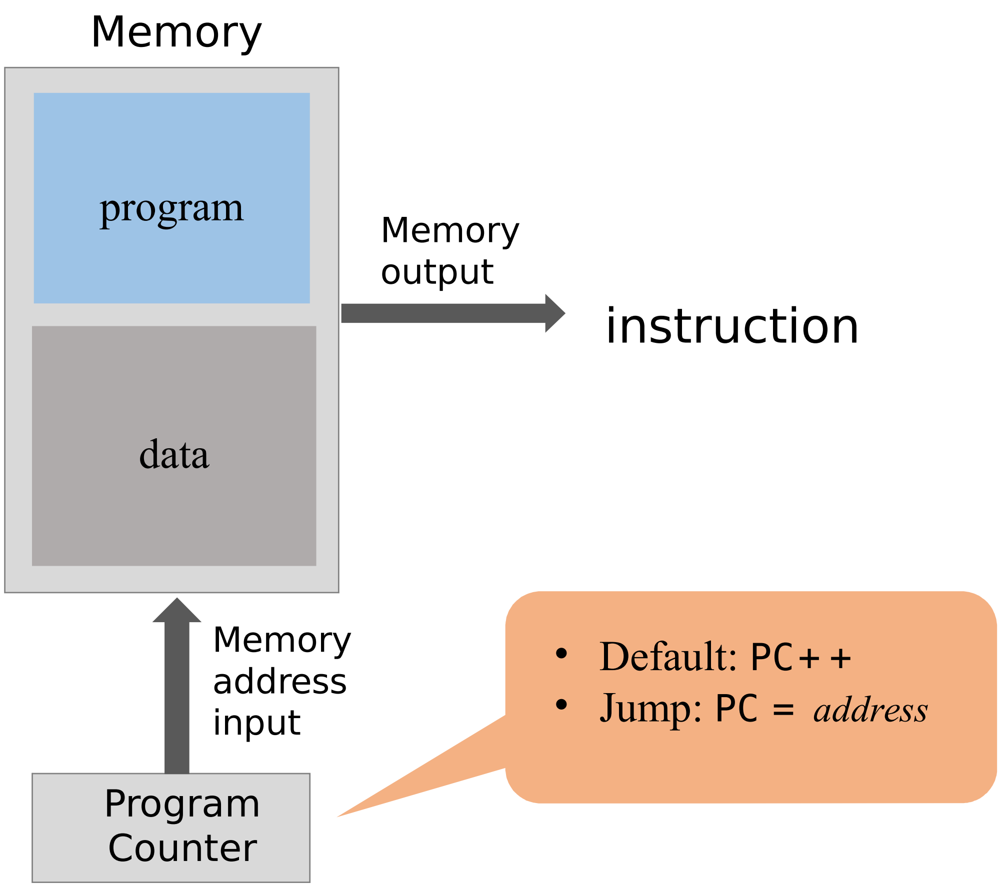
Basic CPU loop

Repeat:

- ***Fetch*** an instruction from the program memory
- ***Execute*** the instruction.

Fetching

- Put the location of the next instruction in the Memory *address* input
- Get the instruction code by reading the contents at that Memory location



Executing

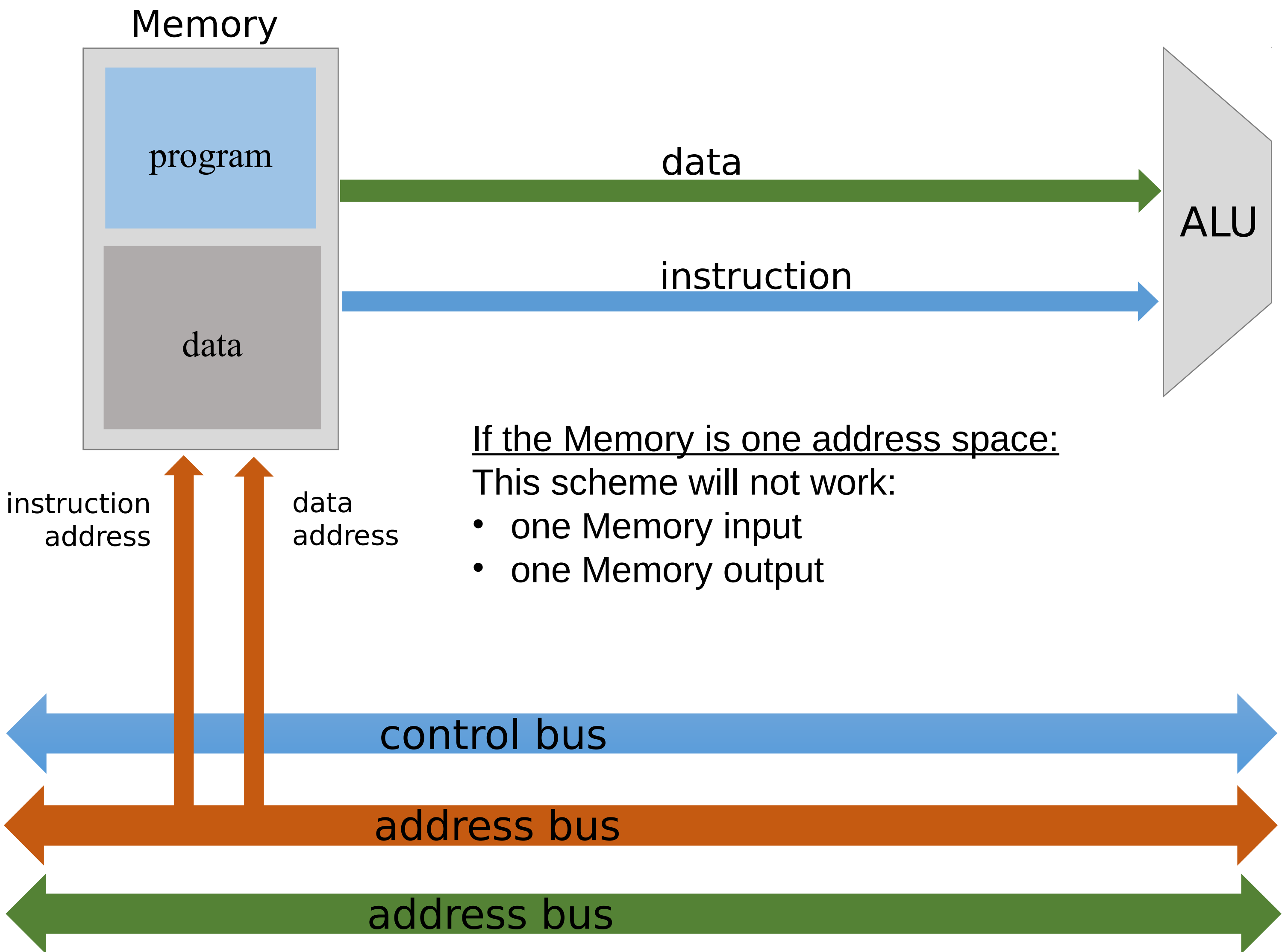
UNIVERSITY OF LEEDS

- The instruction code specifies “what to do”
  - Which arithmetic or logical instruction to execute
  - Which memory address to access (for read / write)
  - If / where to jump
  - …

different subsets of the instruction bits control different aspects of the operation

- Executing the instruction involves:
  - accessing registers
  - and / or:
  - accessing the data memory.

Fetch-Execute Clash

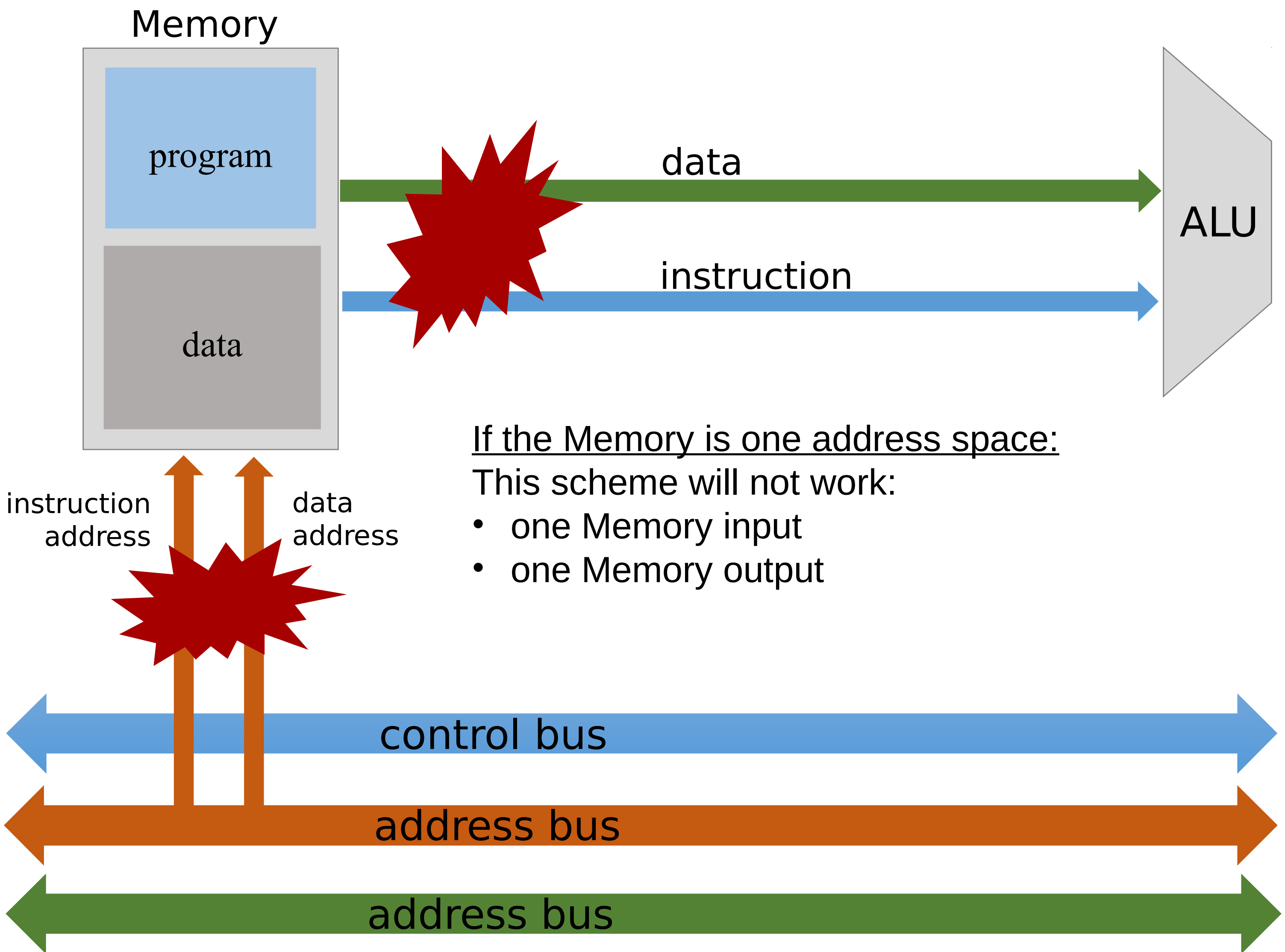


If the Memory is one address space:

This scheme will not work:

- one Memory input
- one Memory output

Fetch-Execute Clash



<u>If the Memory is one address space:</u>

This scheme will not work:

- one Memory input
- one Memory output

Simpler Solution: Seperate Memory Units

UNIVERSITY OF LEEDS

Variant of von Neumann Architecture (used by the Hack computer):

- Two physically separate memory units:
  - Instruction memory
  - Data memory

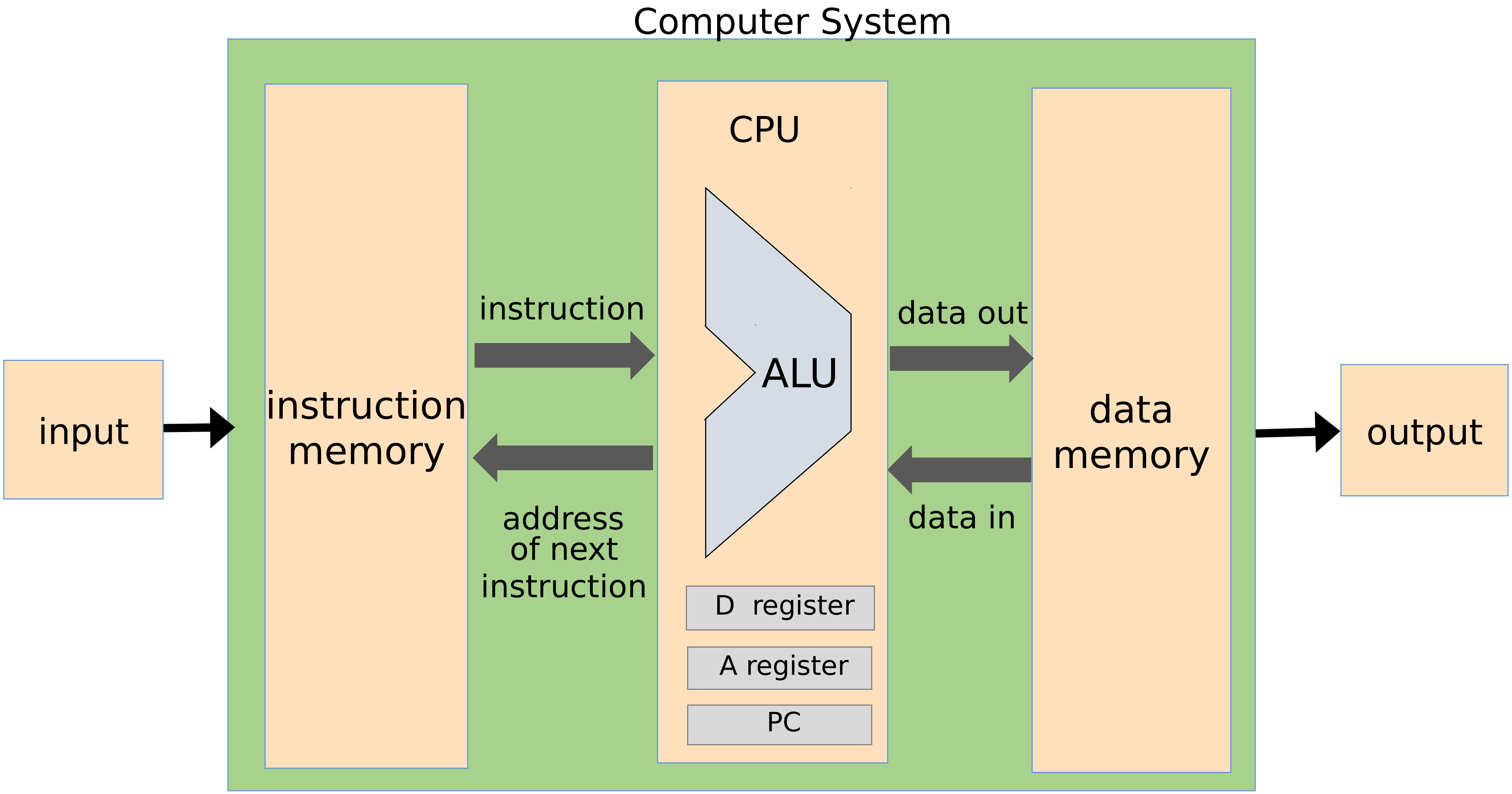
  Each can be addressed and manipulated seperately, and simultaneously

- Advantage:
  - Complication avoided

Sometimes called "Harvard Architecture"

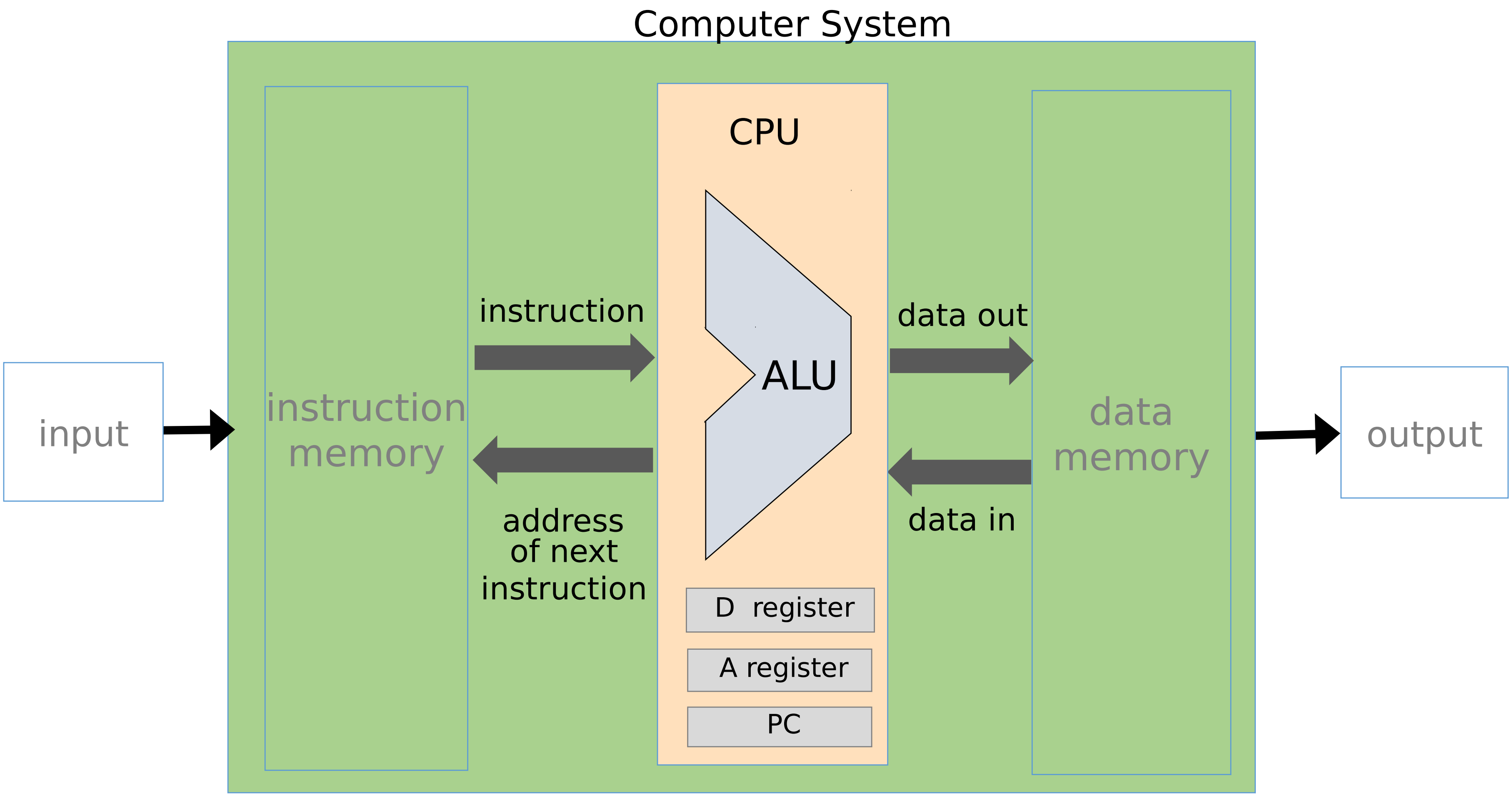
- Disadvantage:
  - Two memory chips instead of one
  - The size of the two chips is fixed.

Hack Computer



Hack CPU

UNIVERSITY OF LEEDS

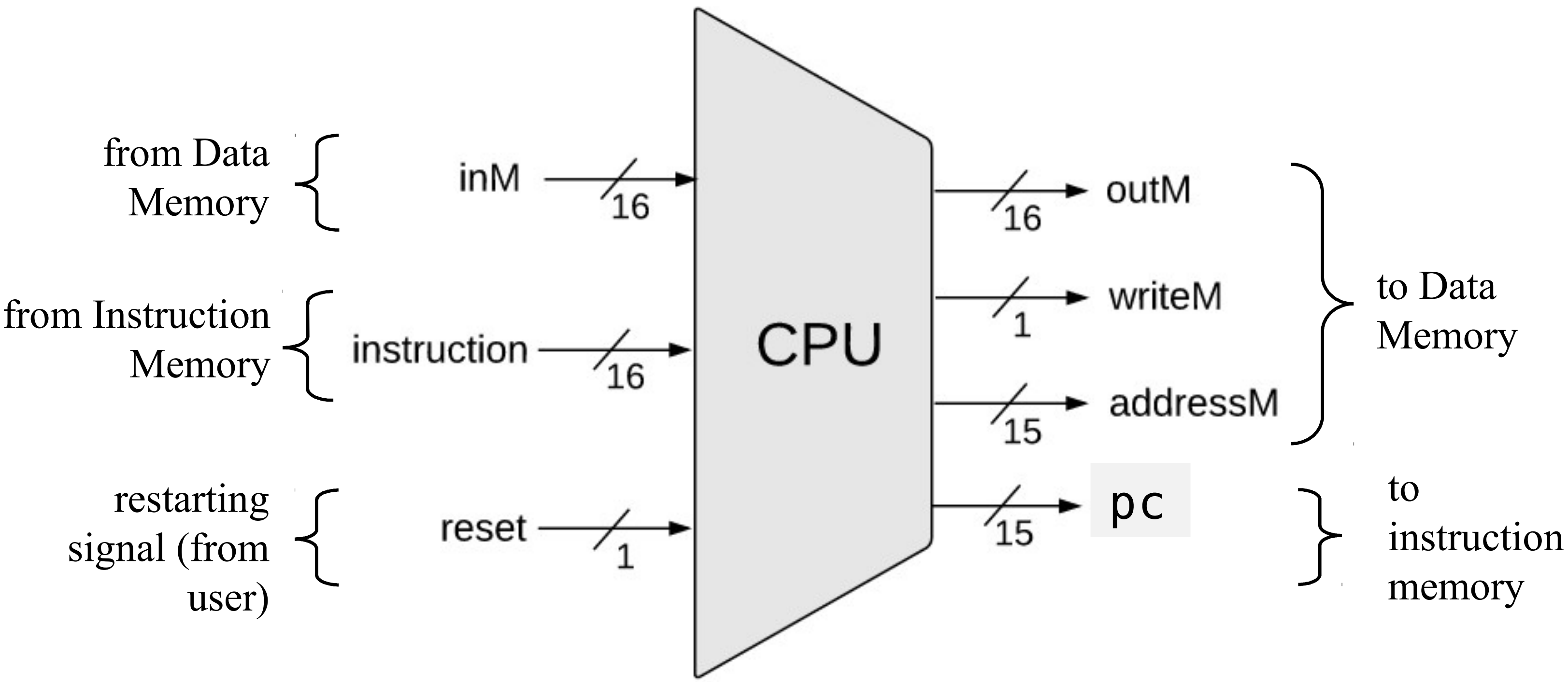


Hack CPU: A 16-bit processor, designed to:

- Execute the current instruction: dataOut = instruction(dataIn)
- Figure out which instruction to execute next.

Hack CPU Interface

UNIVERSITY OF LEEDS



Inputs:

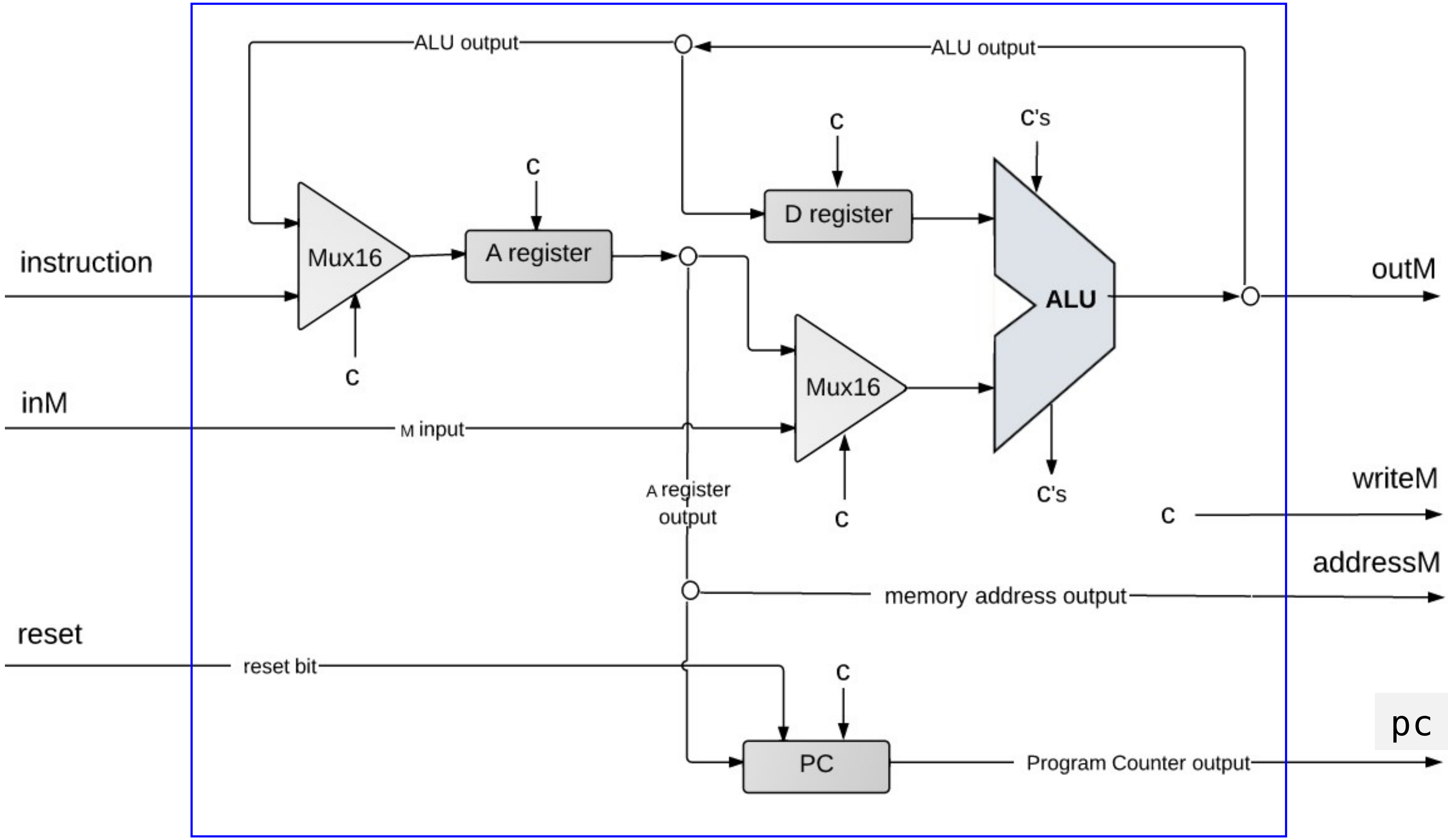
- Data value
- Instruction
- Reset bit

Outputs:

- Data value
- Write to memory (yes/no)
- Memory address
- Adress of next instruction

Hack CPU Implementation

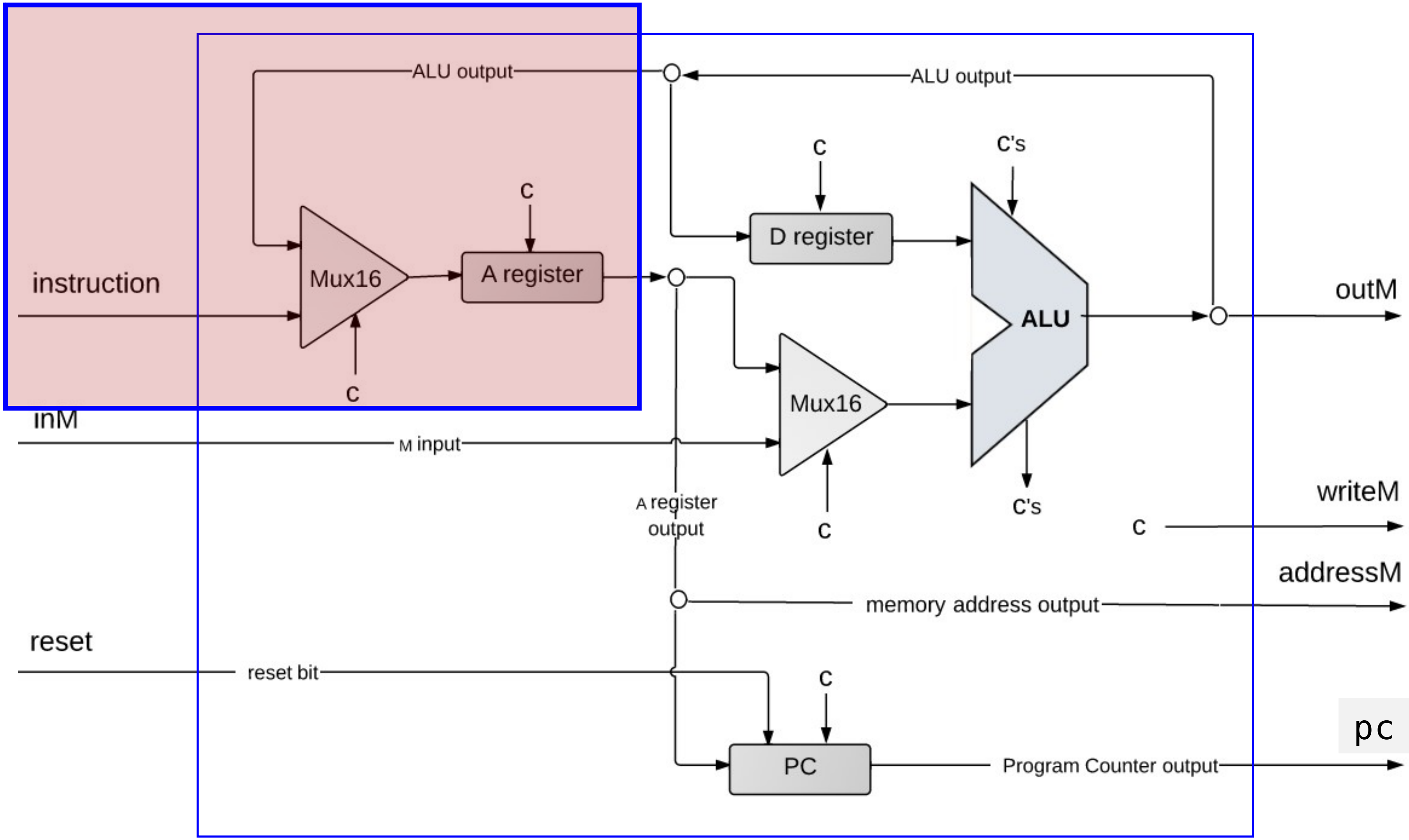
UNIVERSITY OF LEEDS



(each "C" symbol represents a control bit)

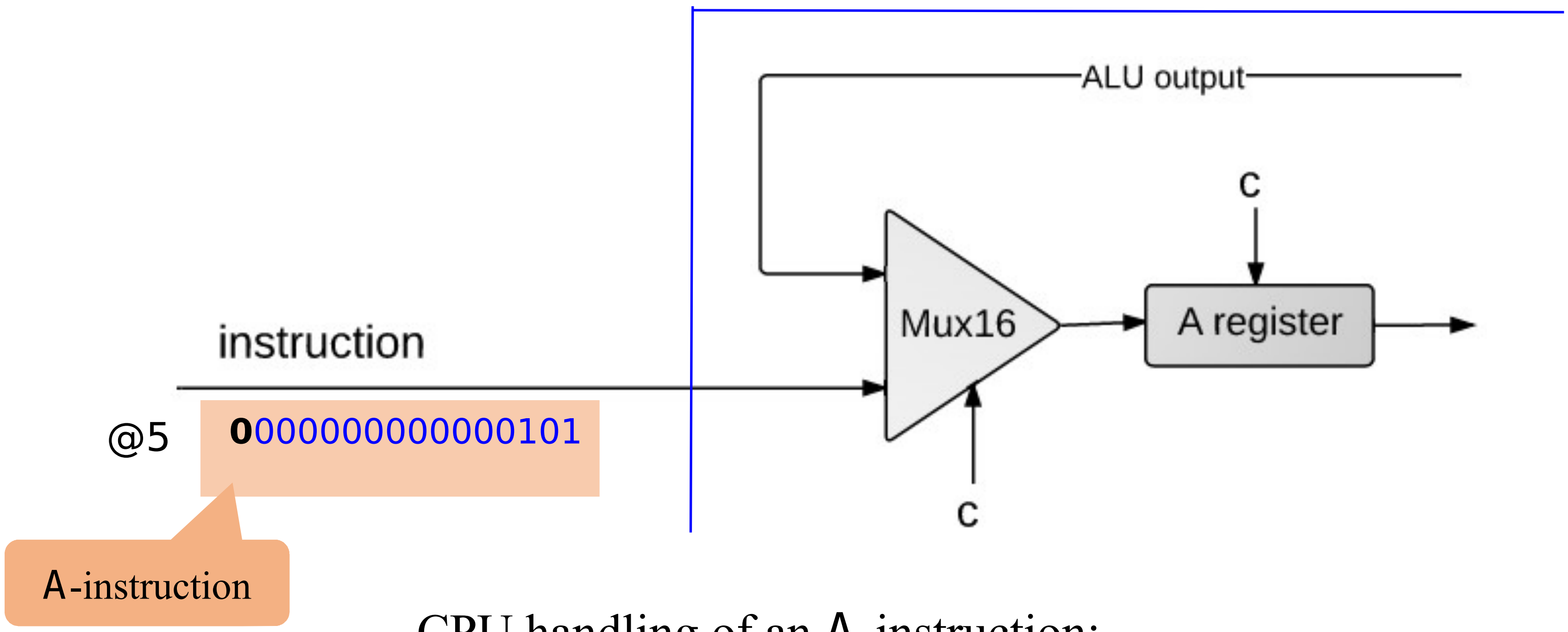
CPU operation: Instruction Handling

UNIVERSITY OF LEEDS



(each "C" symbol represents a control bit)

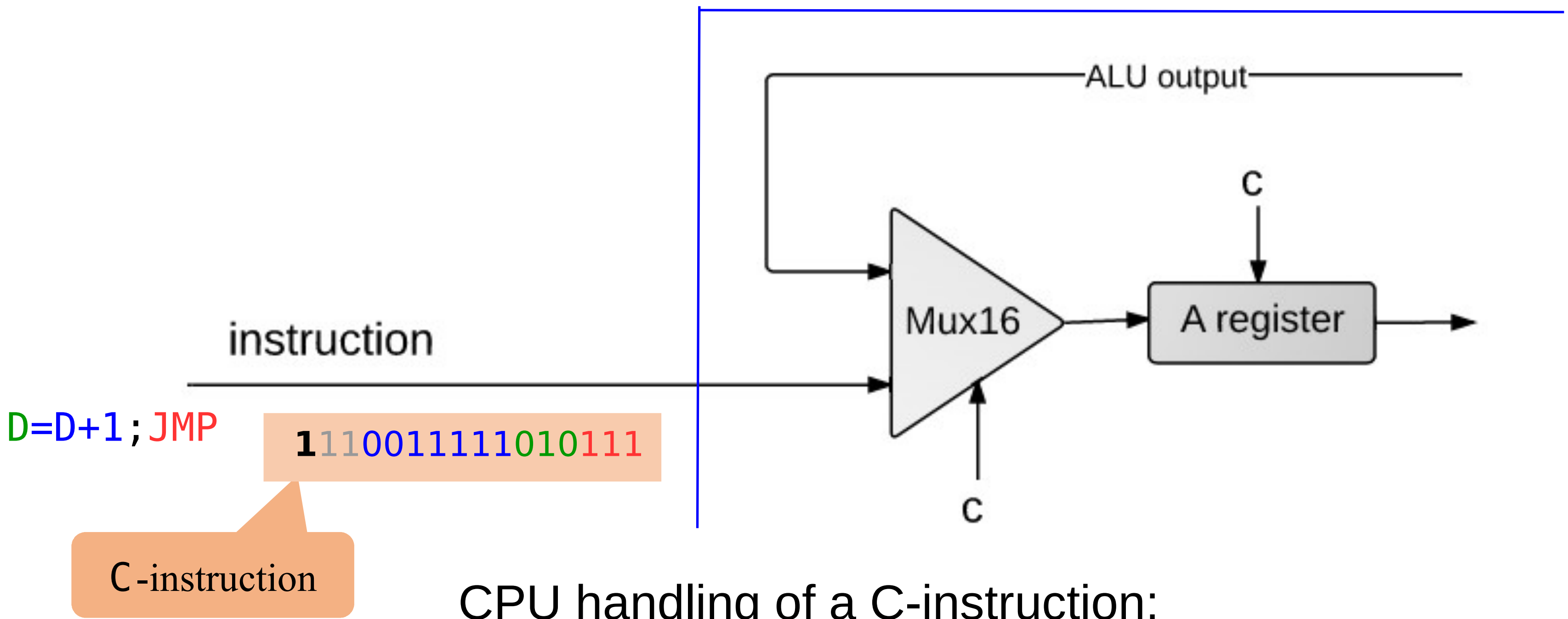
CPU operation: Instruction Handling



<u>CPU handling of an A-instruction:</u>

- 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

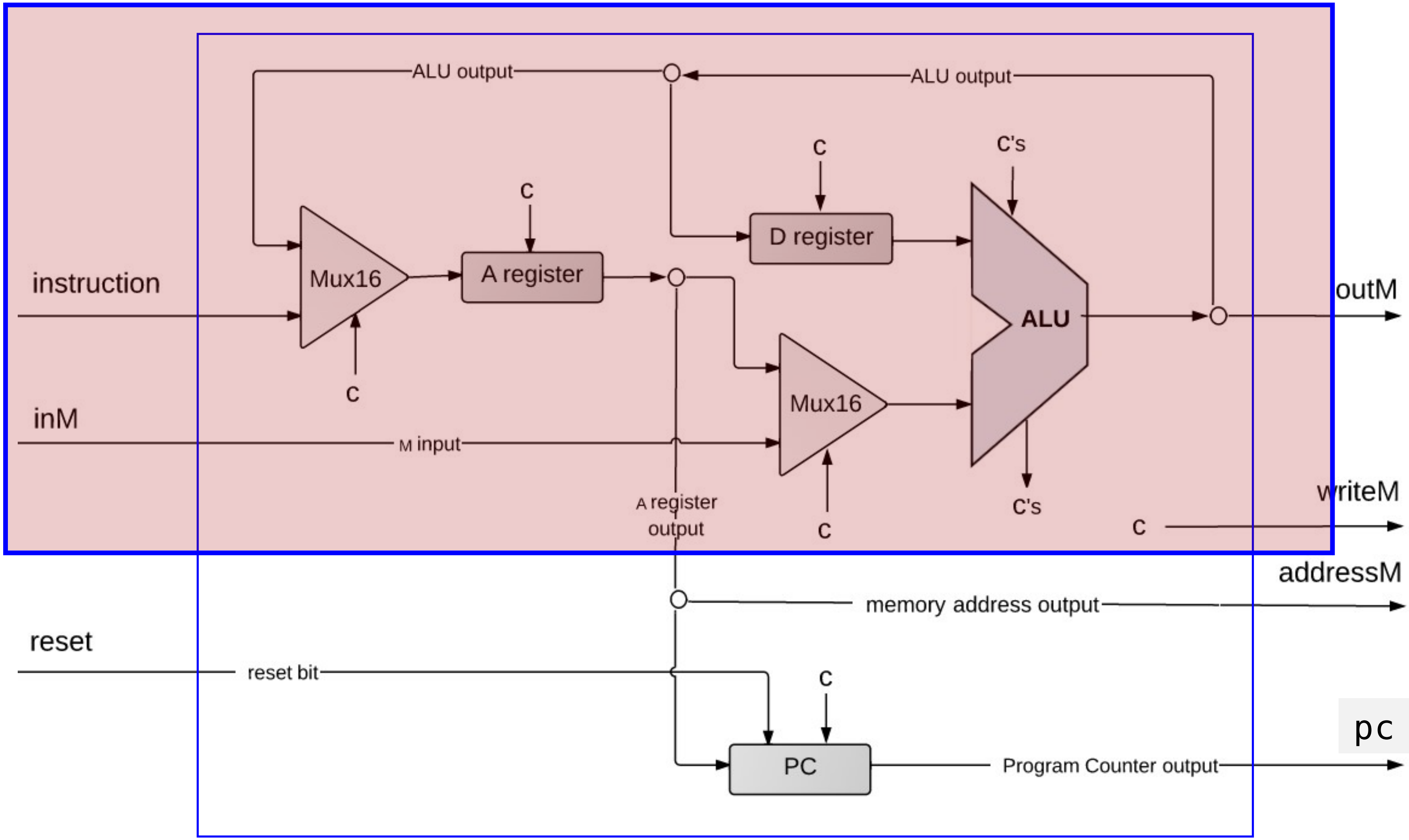


CPU handling of a C-instruction:

- 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.

ALU operation

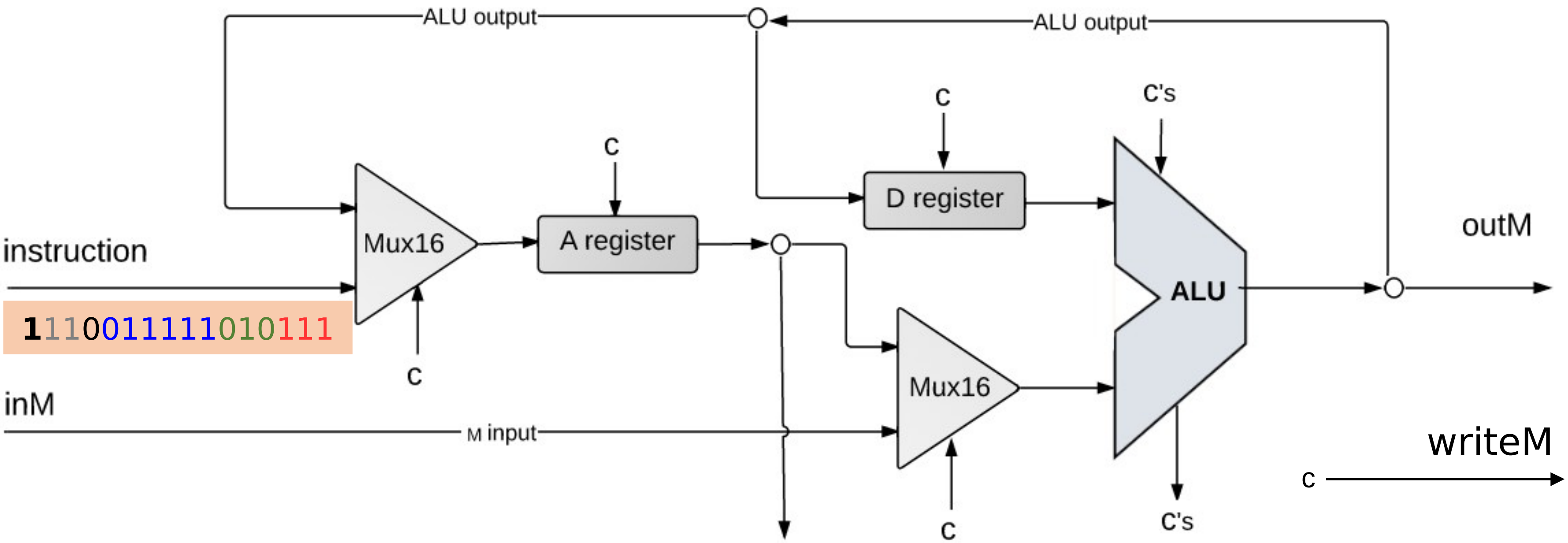
UNIVERSITY OF LEEDS



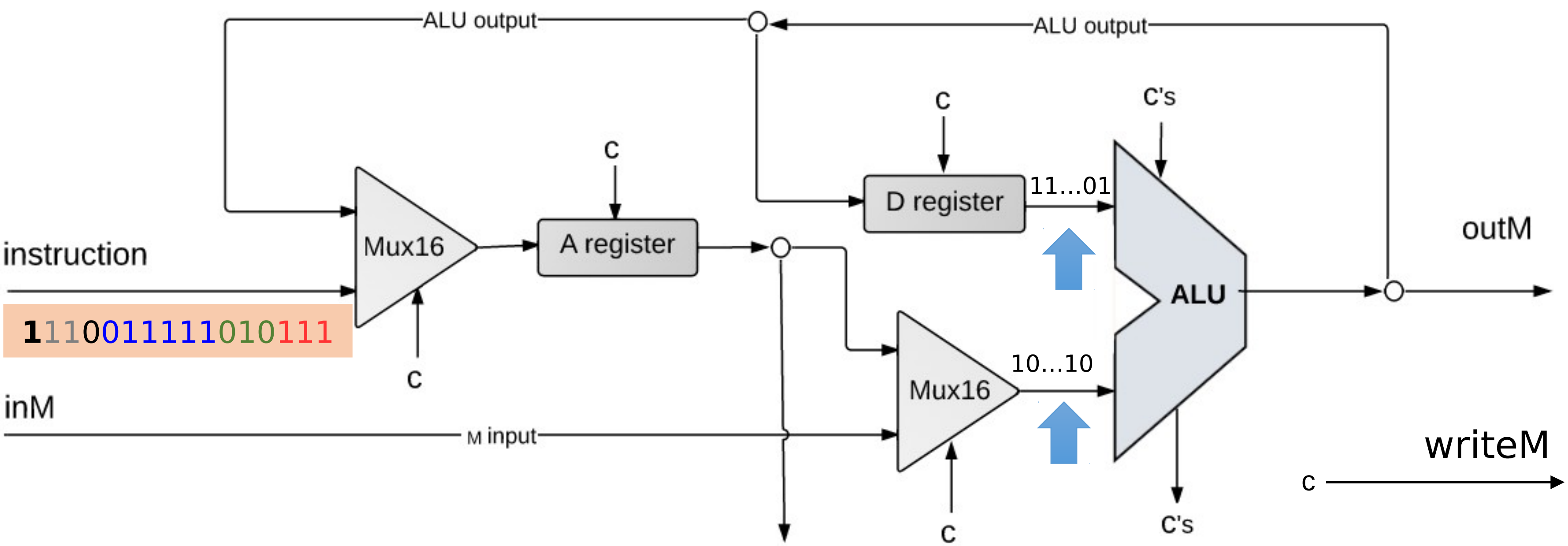
(each "C" symbol represents a control bit)

ALU operation: inputs

UNIVERSITY OF LEEDS



ALU operation: inputs

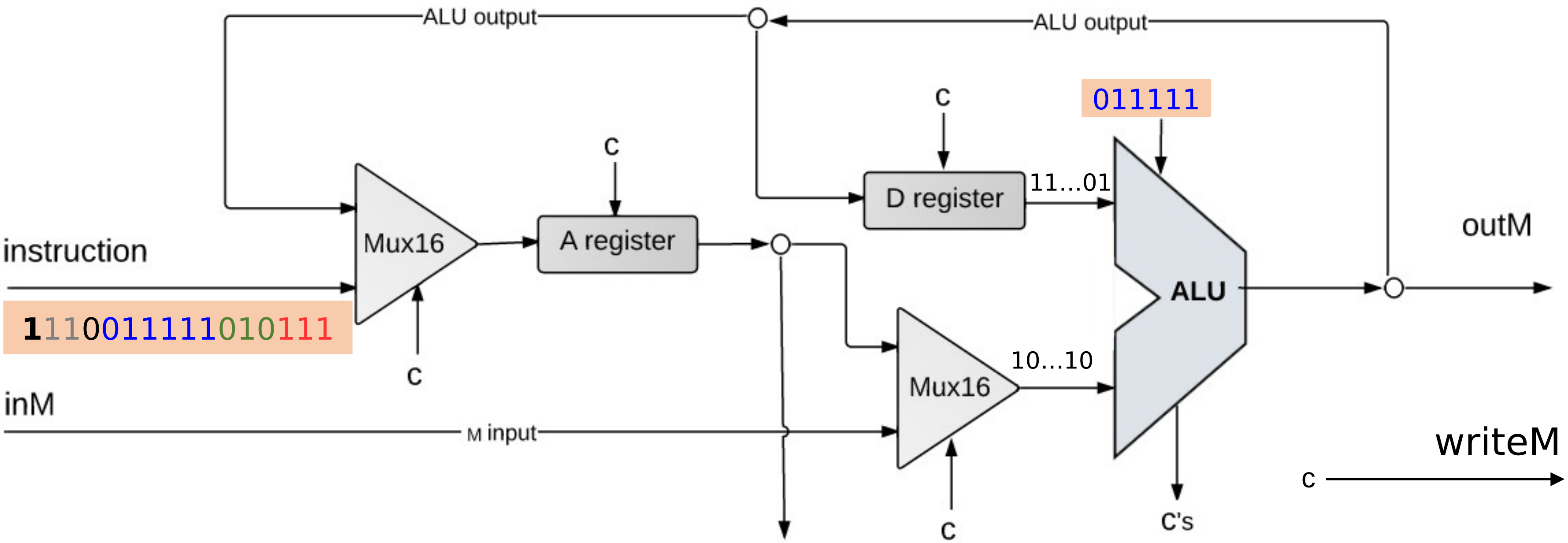


<u>ALU data inputs:</u>

- from the `D-register`
- from either:
  - `A-register`, or
  - data memory (`M-register`)

ALU operation: inputs

UNIVERSITY OF LEEDS



ALU data inputs:

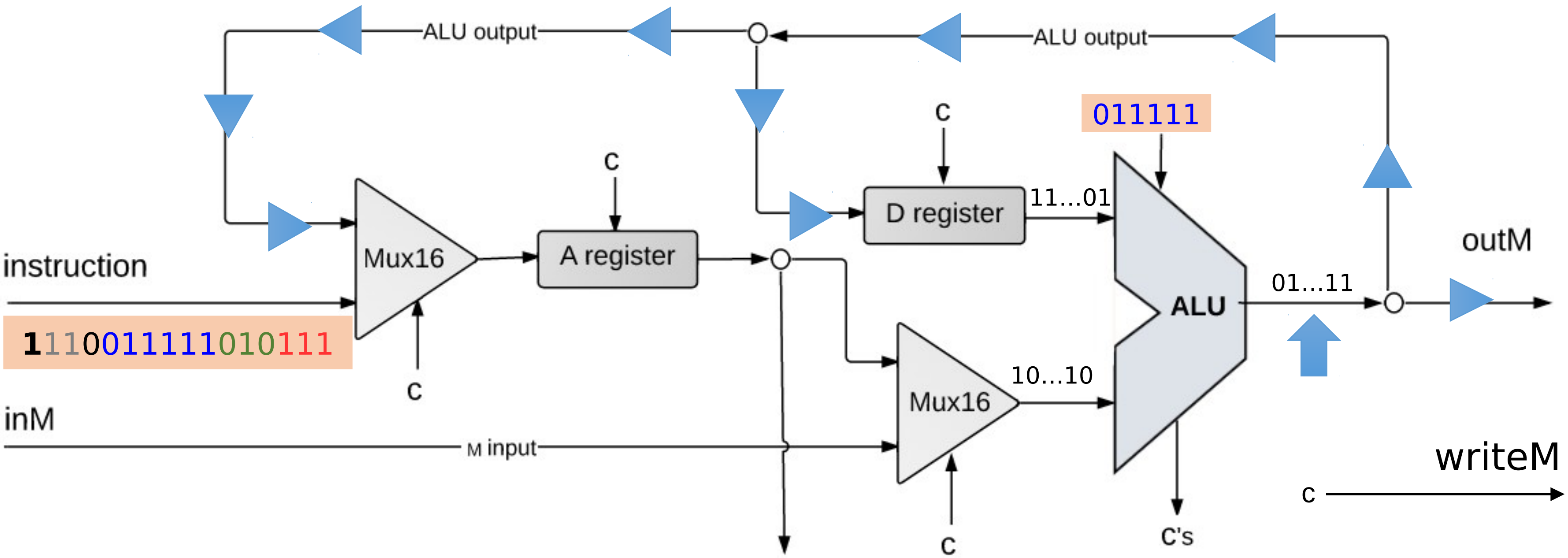
- from the `D-register`
- from either:
  - `A-register`, or
  - data memory (`M-register`)

ALU control inputs:

- control bits
  (from the instruction)

ALU operation: outputs

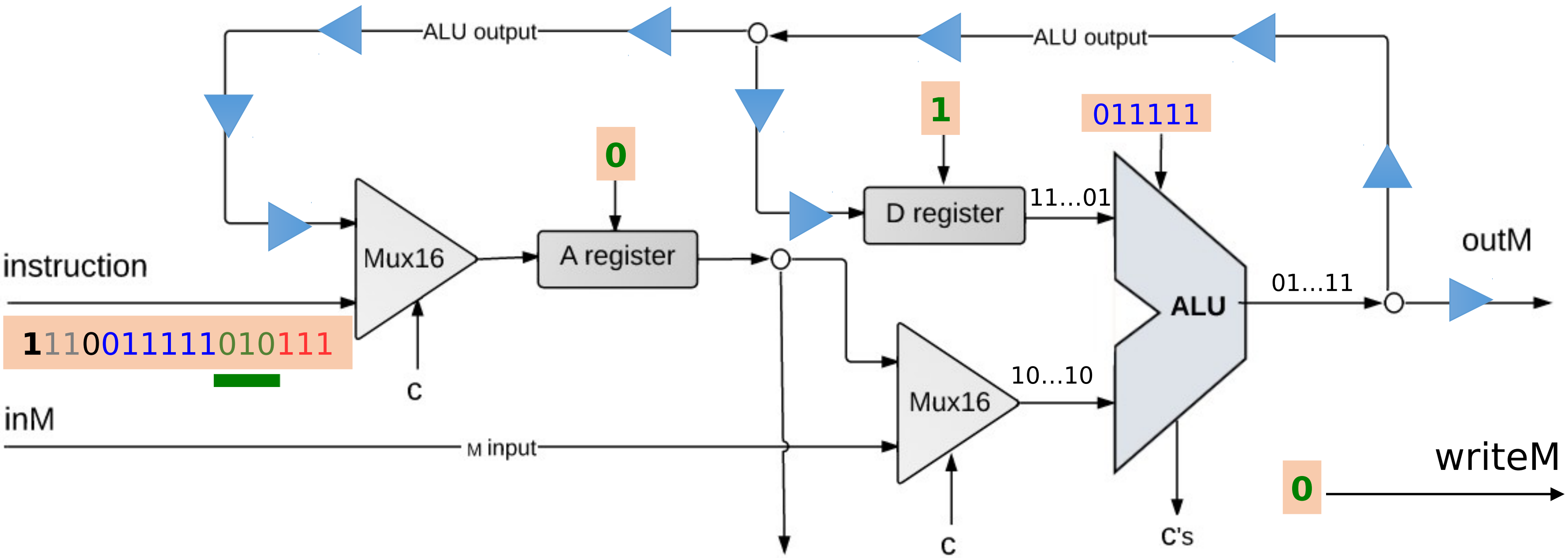
UNIVERSITY OF LEEDS



ALU data output

- Result of ALU calculation fed simultaneously to: `D-register`, `A-register`, `M-register` (data memory)

ALU operation: outputs

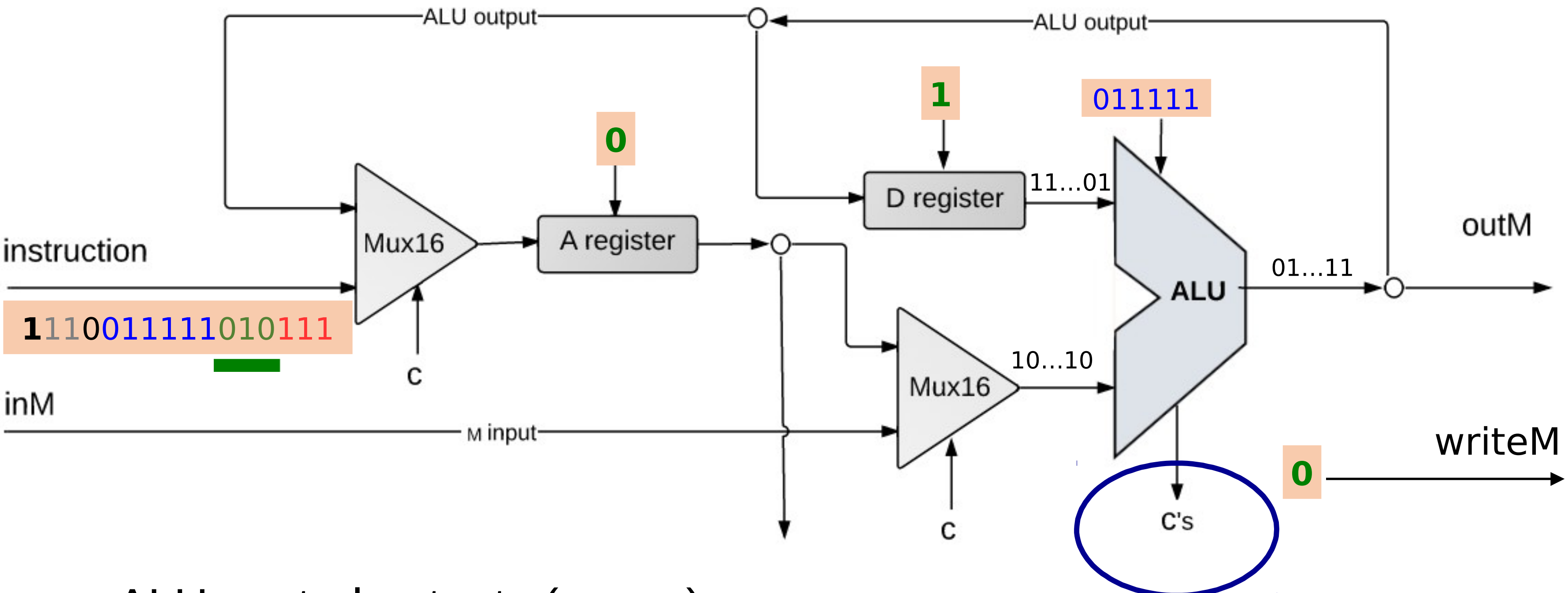


<u>ALU data output</u>

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

ALU operation: outputs

UNIVERSITY OF LEEDS



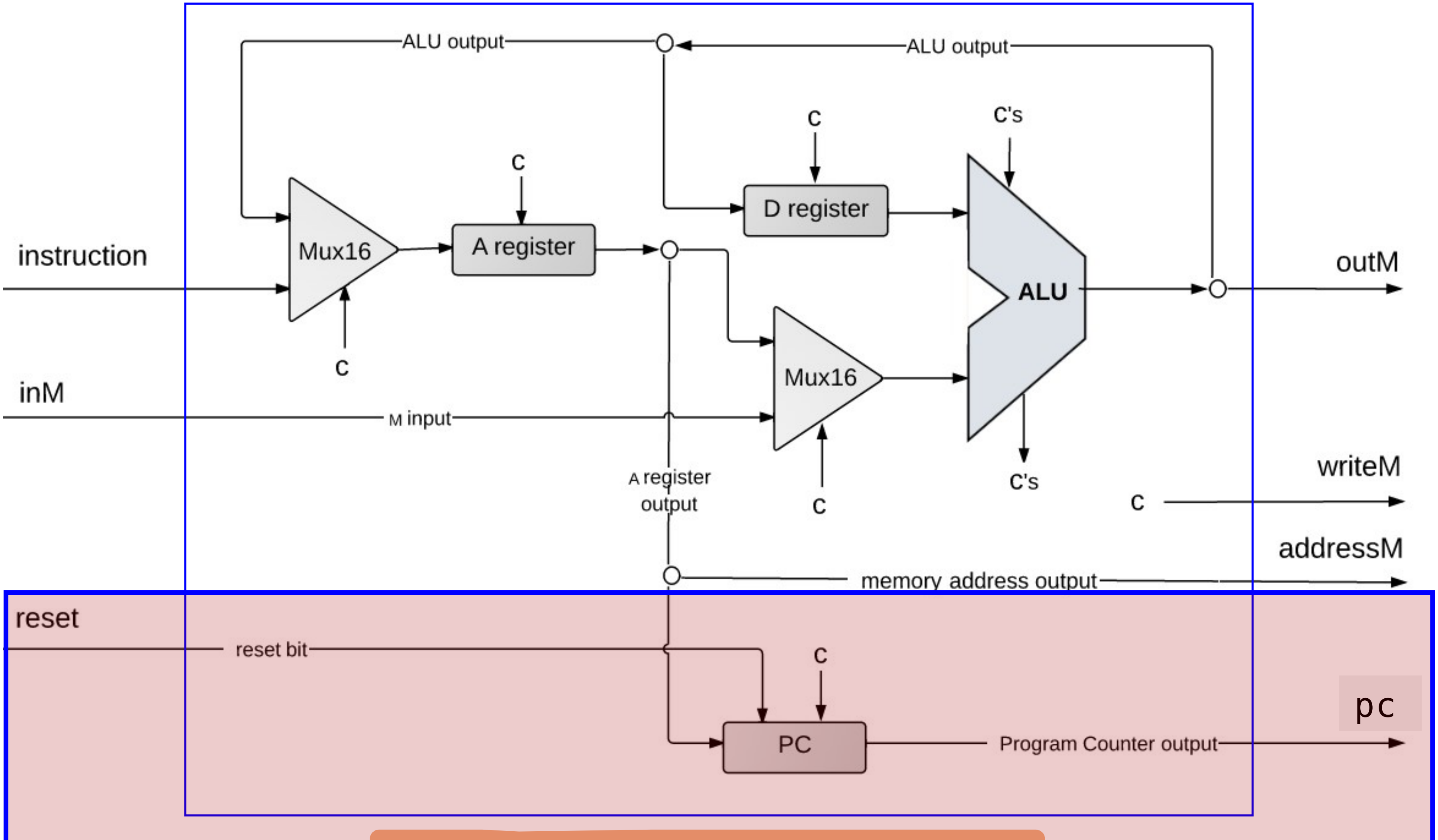
ALU control outputs (zr,ng):

- Is the output negative?

  `ng; // 1 if (out < 0),  0 otherwise`
- Is the output zero?

  `zr; // 1 if (out == 0), 0 otherwise`

CPU operation: control



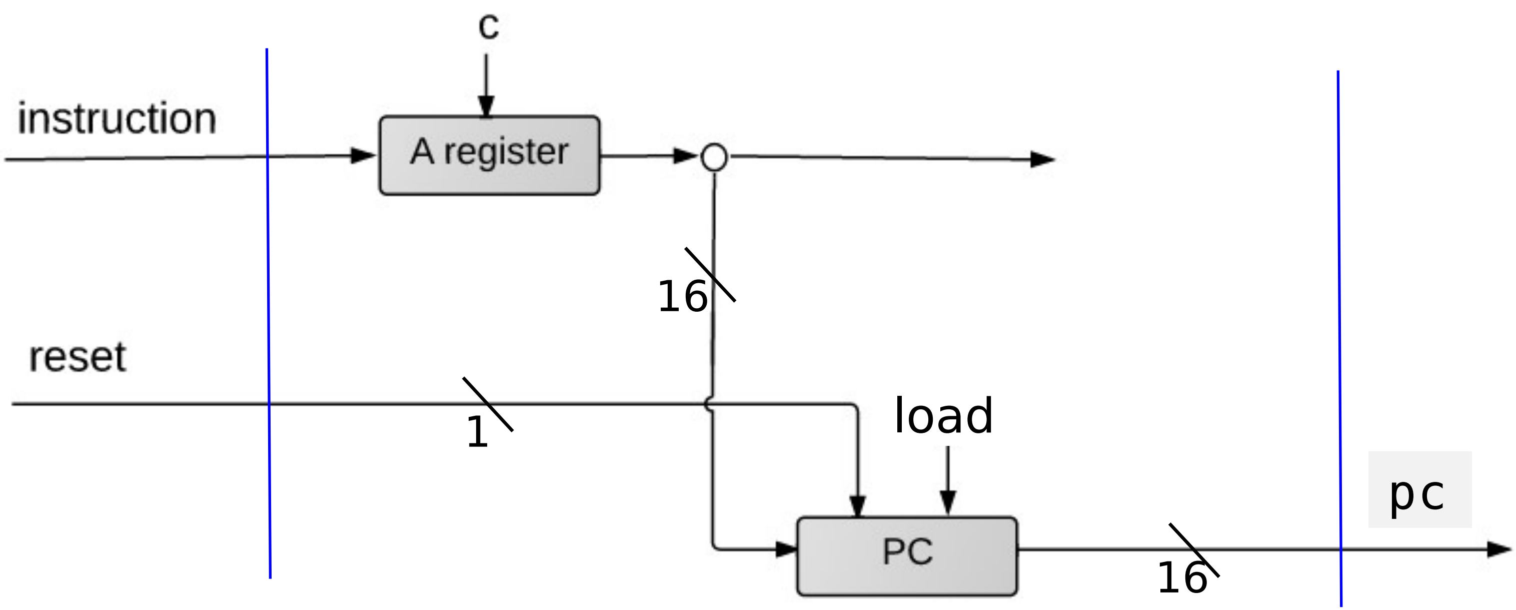
(each "C" symbol represents a control bit)

CPU operation: control

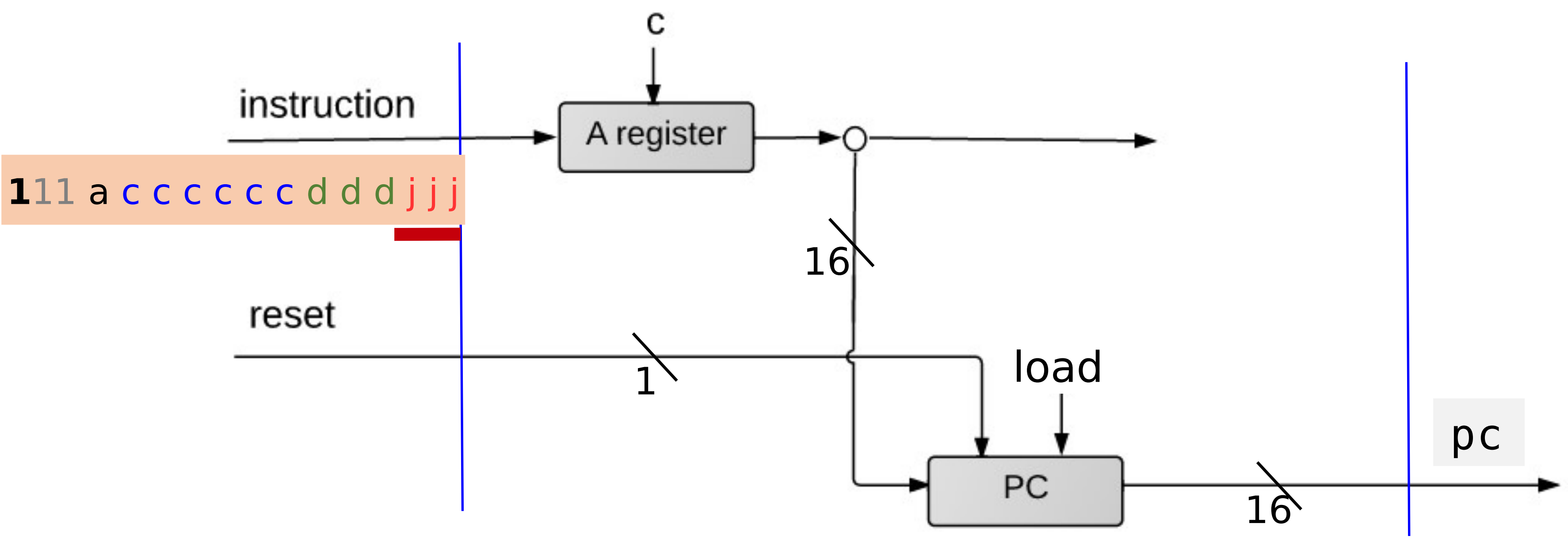
UNIVERSITY OF LEEDS

- The computer is loaded with some program (written in Hack ML);
- Pushing **reset** causes the program to start running.

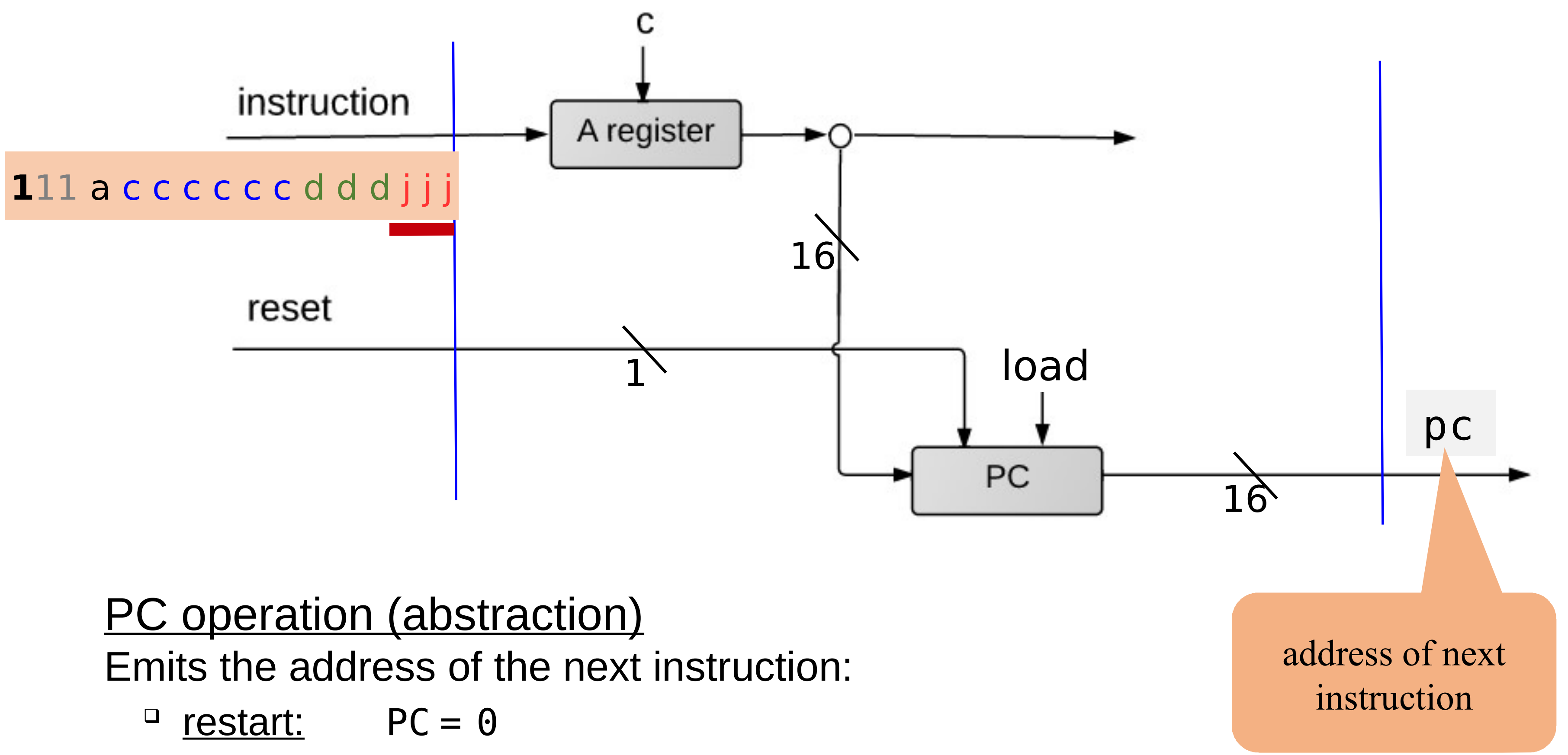
CPU operation: control



CPU operation: control



CPU operation: control



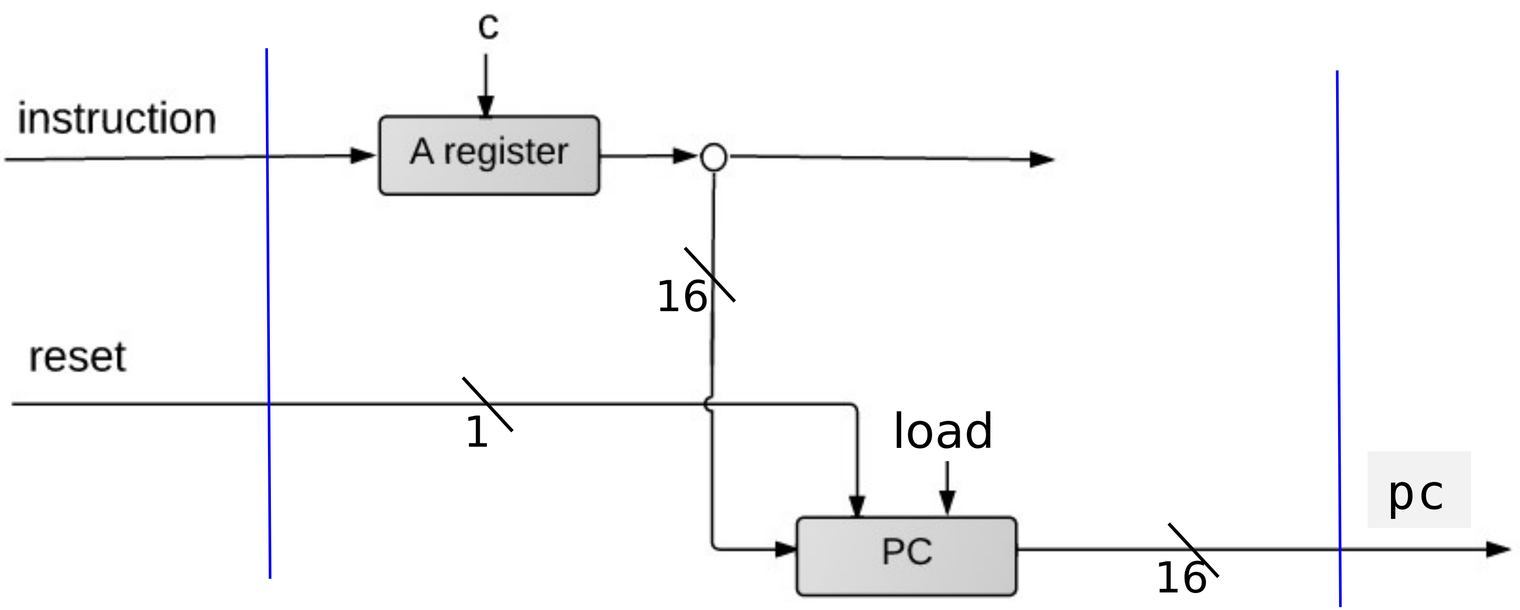
PC operation (abstraction)

Emits the address of the next instruction:

- restart: PC = 0
- no jump: PC++
- goto: PC = A
- conditional goto: if (*condition*) PC = A else PC++

CPU operation: control

UNIVERSITY OF LEEDS

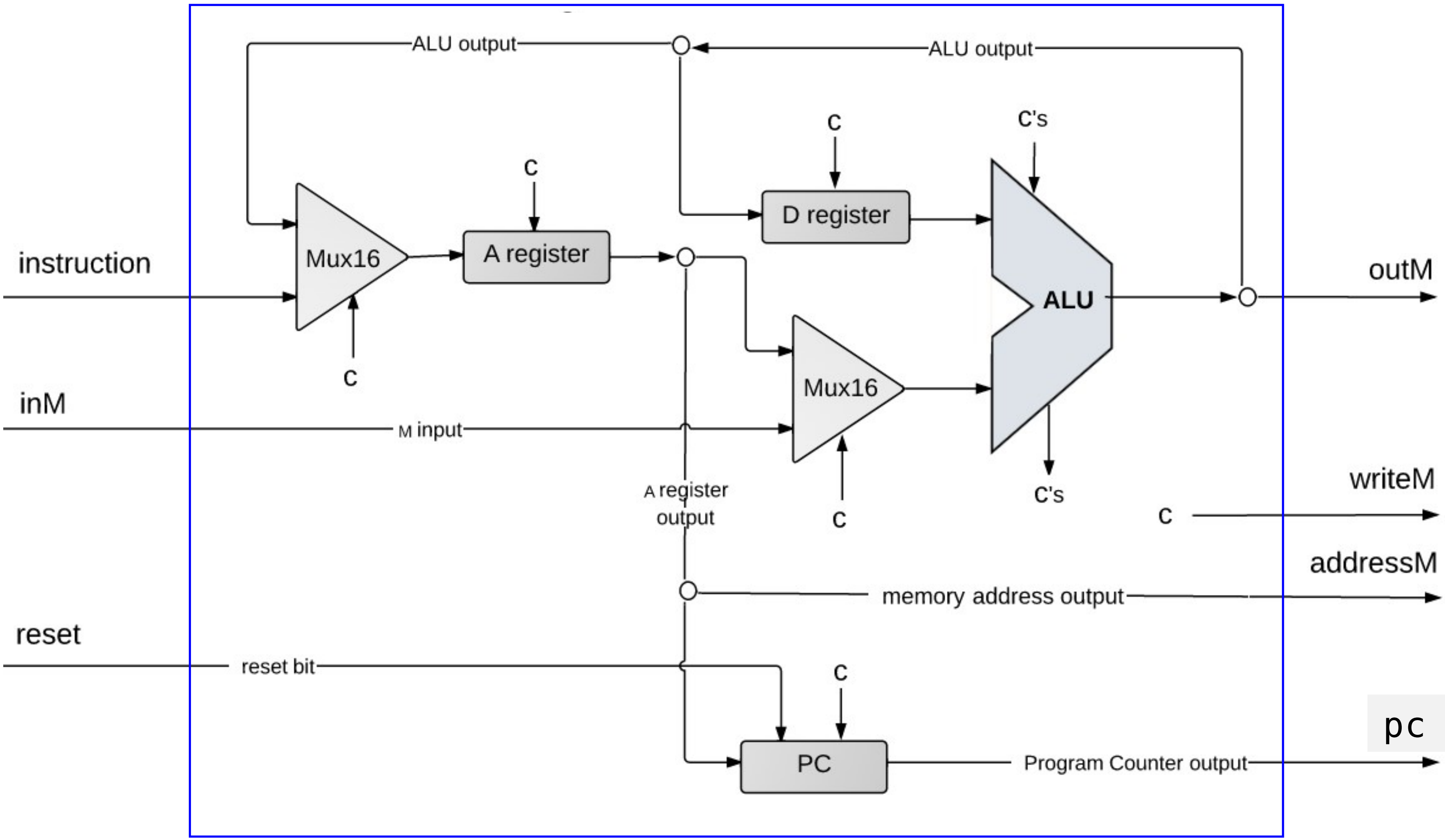


<u>PC operation (implementation)</u>

```
if (reset==1) PC = 0
else
    // current instruction:
        load = f(jump bits, ALU control outputs)
        if (load == 1)  PC = A  // jump
        else            PC++  // next instruction
```

Hack CPU implementation

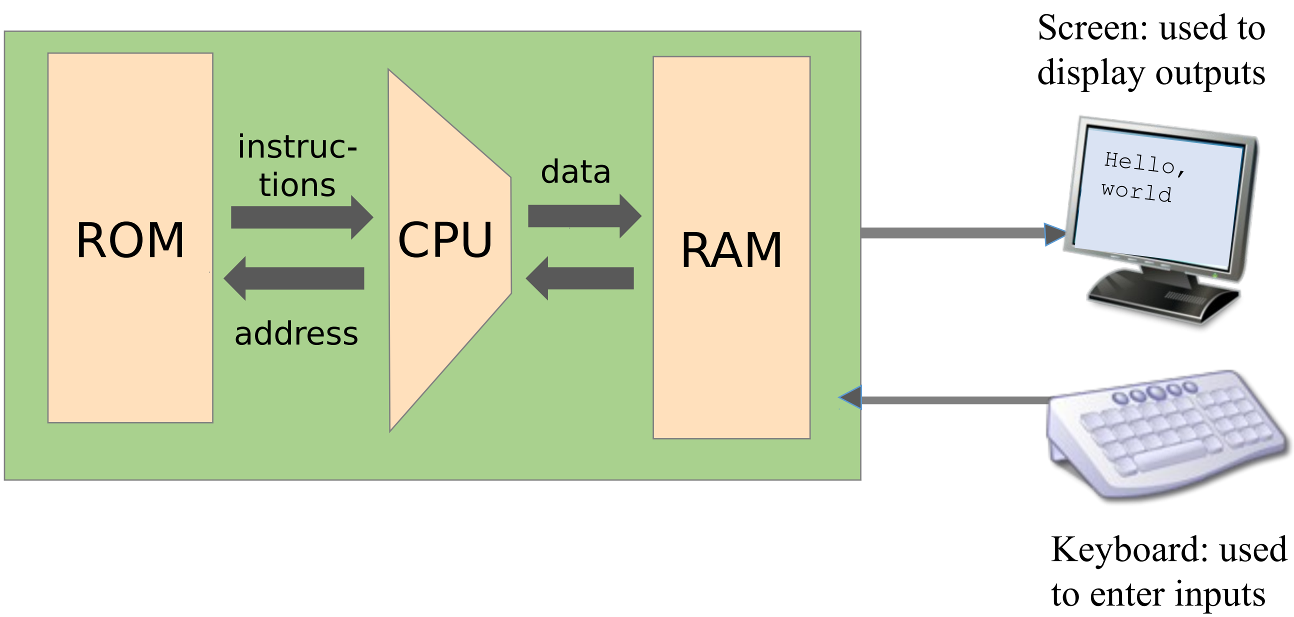
UNIVERSITY OF LEEDS



That's It!

Hack Computer

UNIVERSITY OF LEEDS

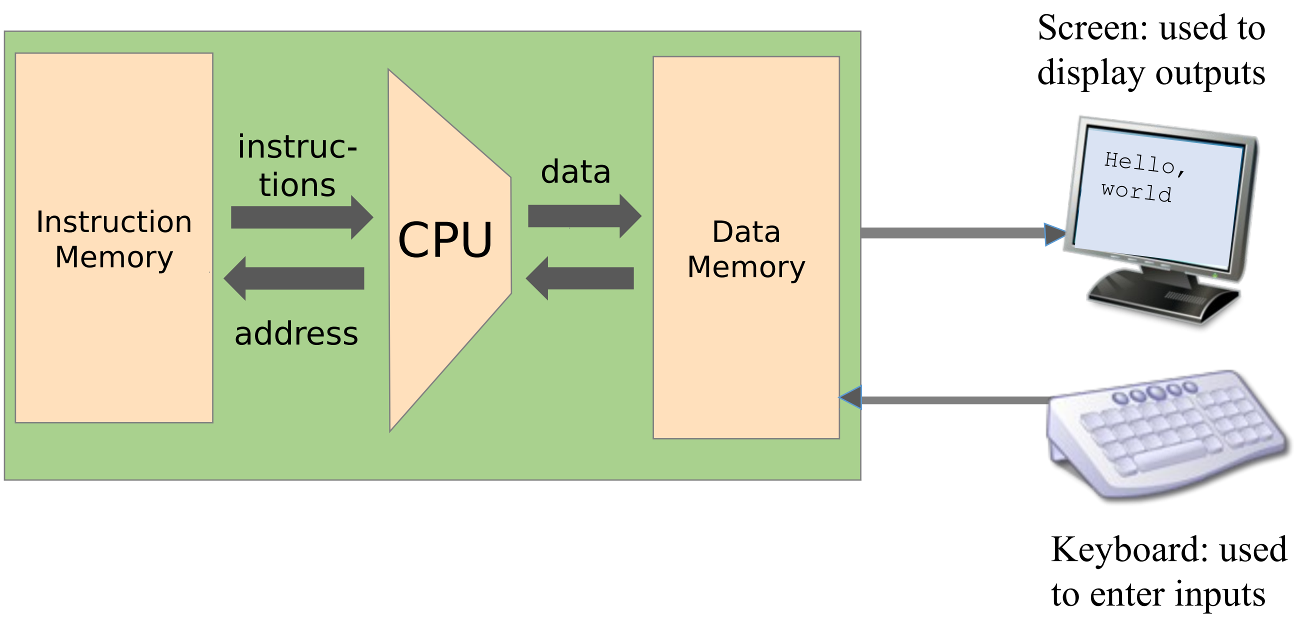


Abstraction:
A computer capable of running programs written in the Hack machine language

Implementation:
Built from the Hack chip-set.

Hack Computer

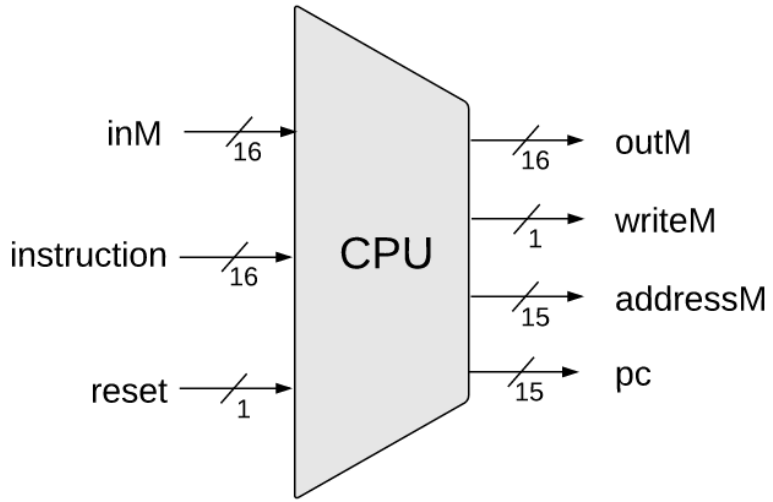
UNIVERSITY OF LEEDS



<u>Abstraction:</u>
A computer capable of running programs written in the Hack machine language

<u>Implementation:</u>
Built from the Hack chip-set.

Hack Computer: CPU

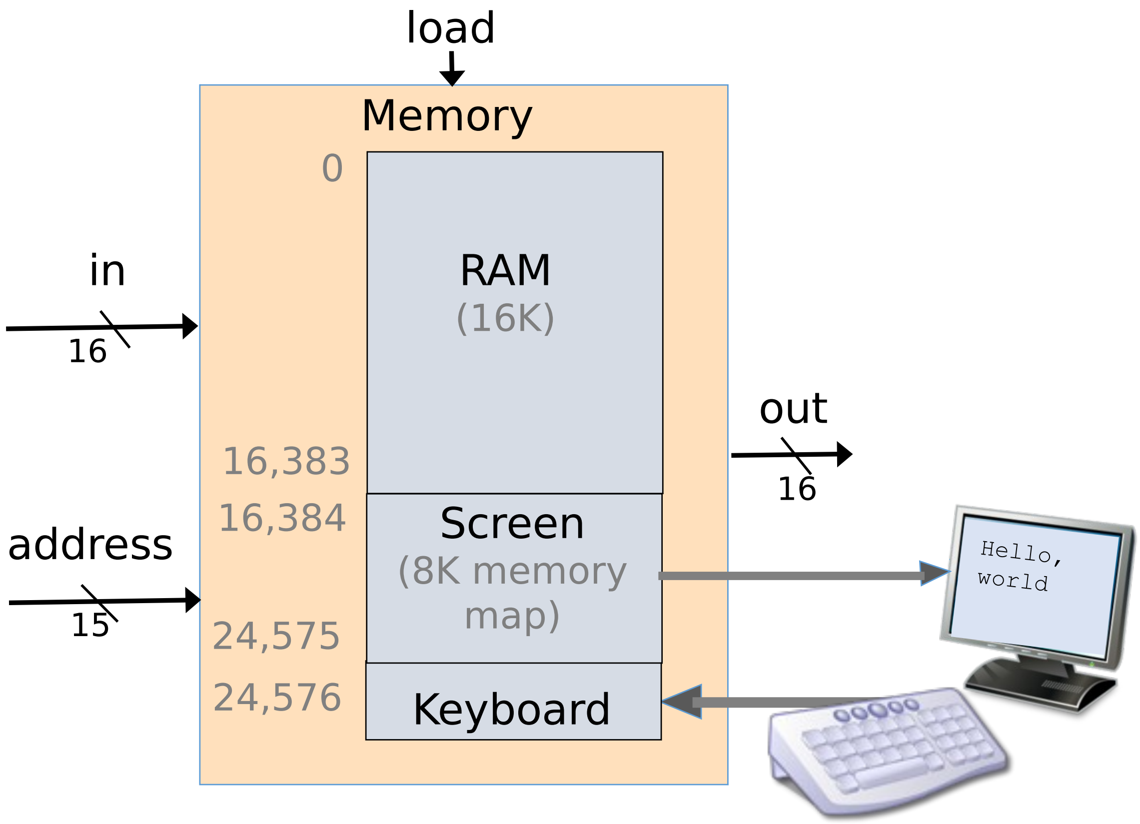


<u>CPU abstraction:</u>
Executes a Hack instruction and figures out which instruction to execute next

<u>CPU Implementation:</u>
Discussed before.

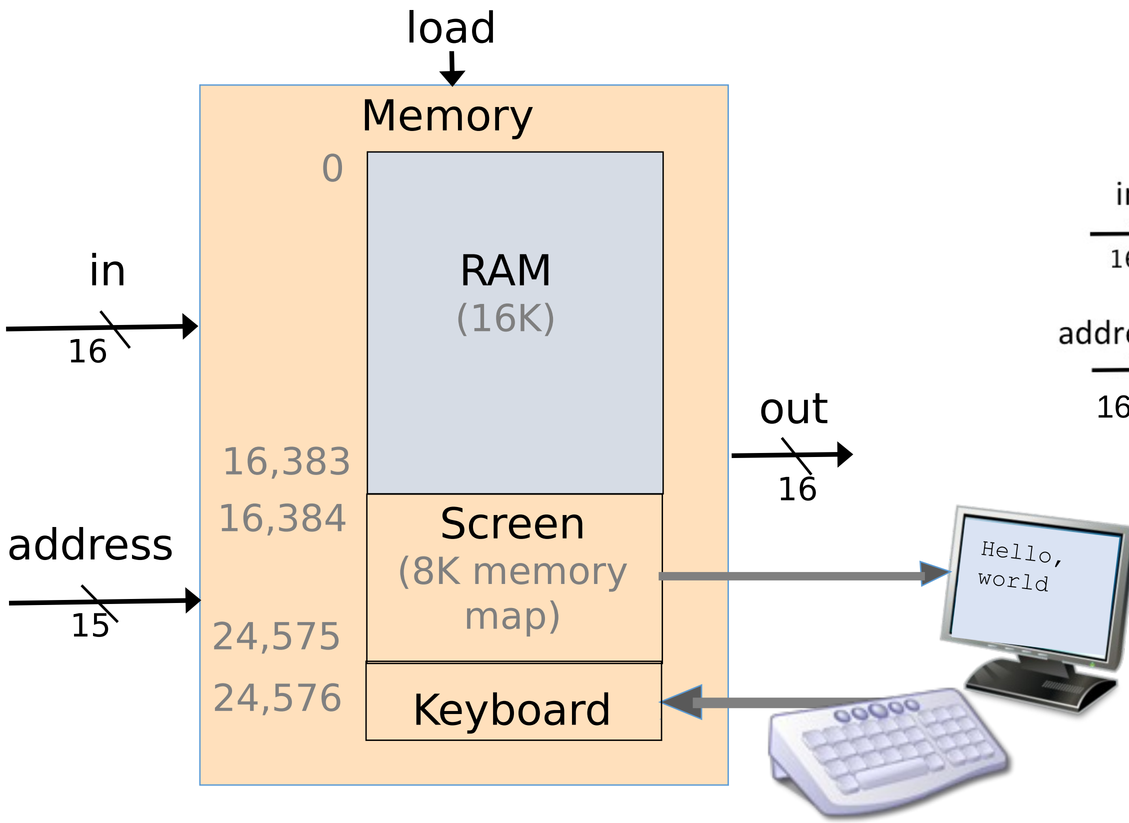
Hack Computer: Data Memory

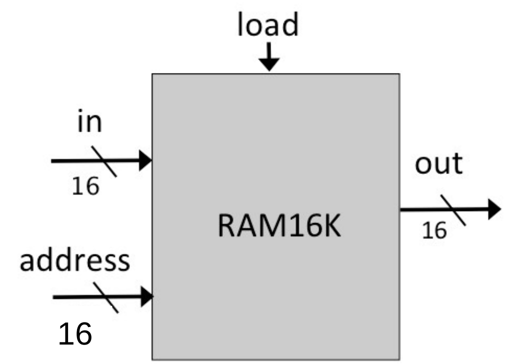
UNIVERSITY OF LEEDS



- Address 0 to 16383: data memory
- Address 16384 to 24575: screen memory map
- Address 24576: keyboard memory map

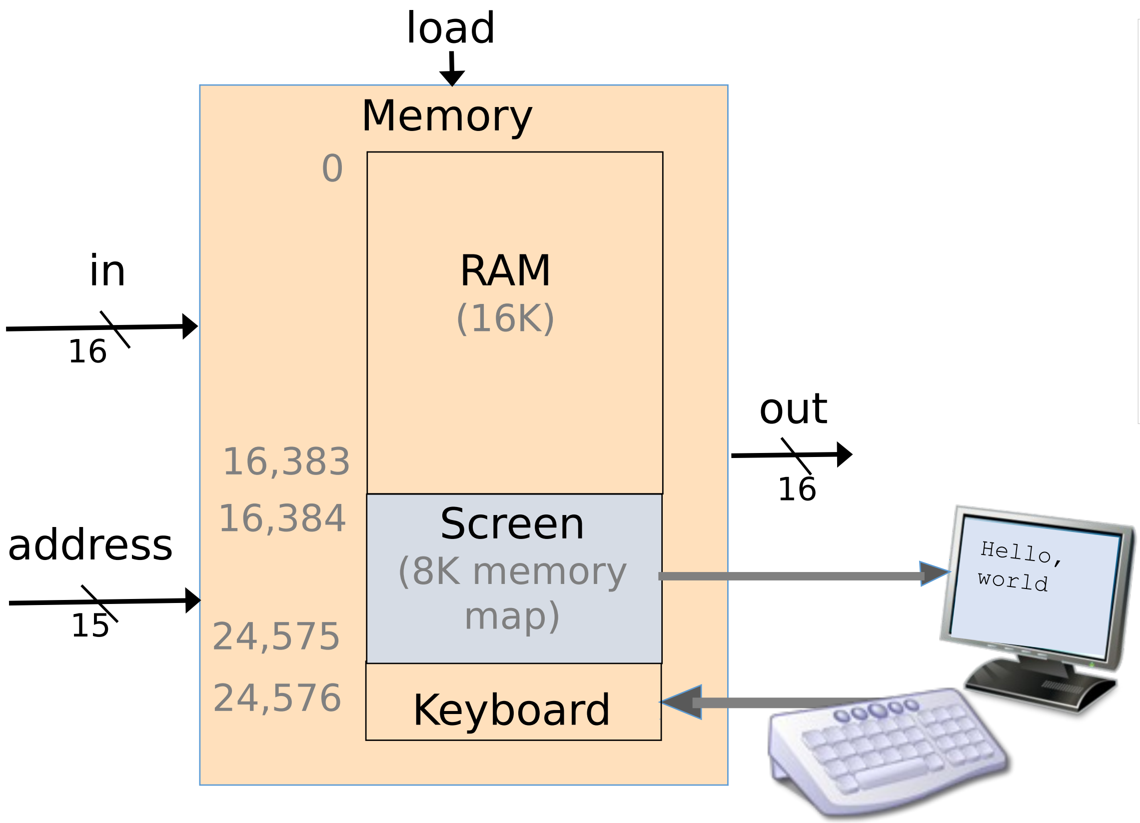
Hack Computer: Data Memory

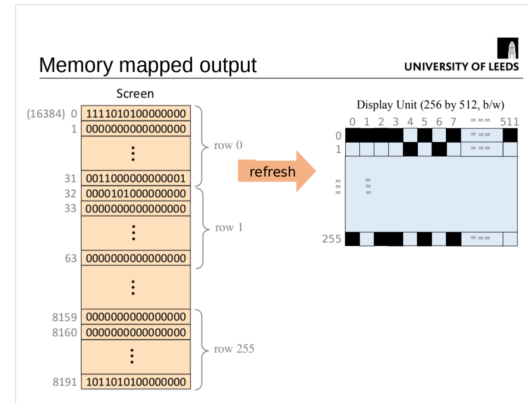




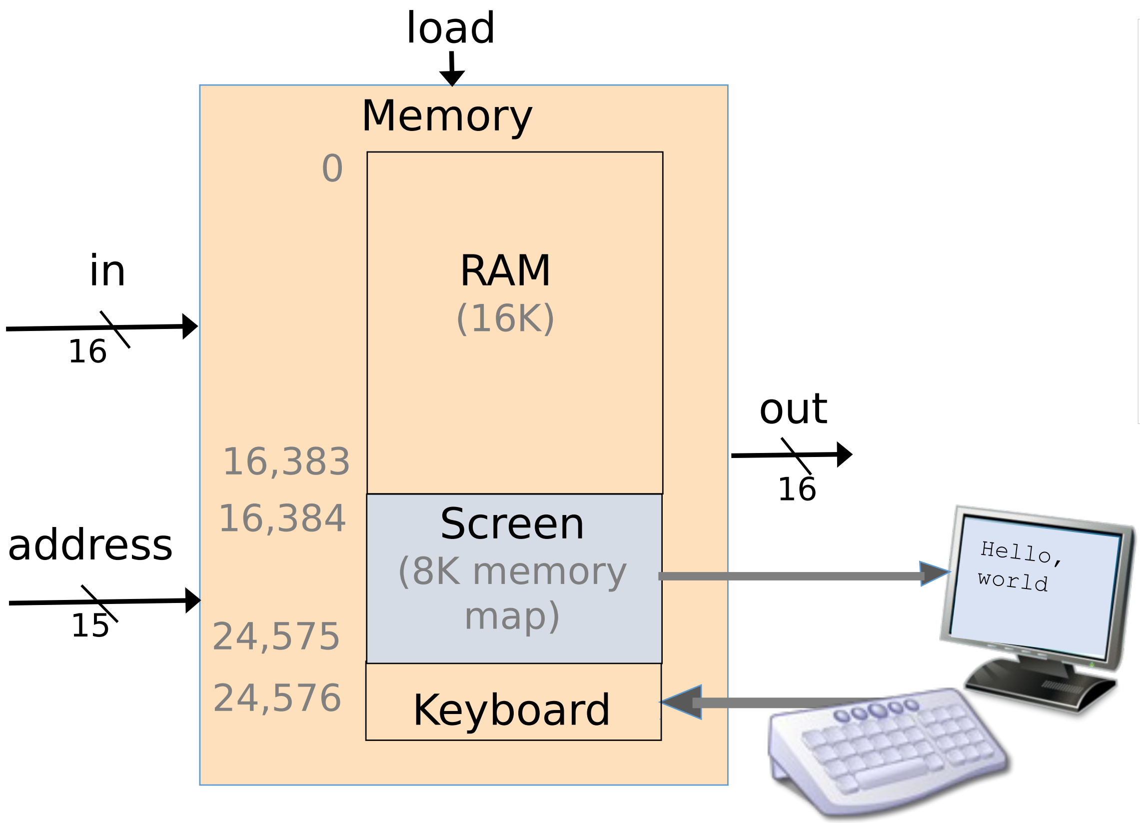
Hack Computer: Data Memory

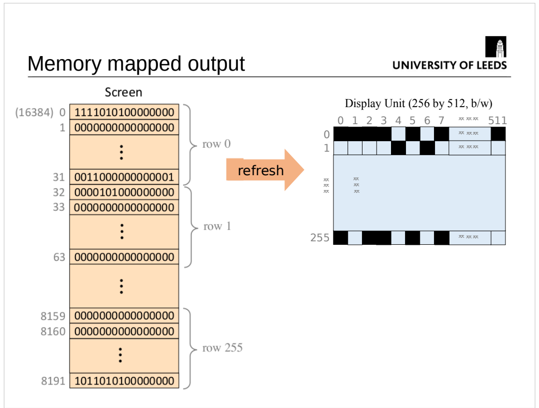
UNIVERSITY OF LEEDS

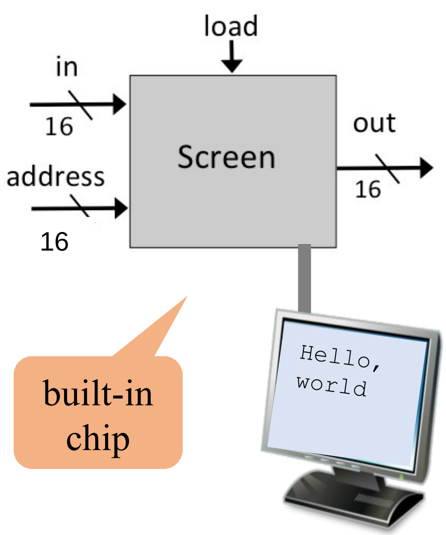




Hack Computer: Data Memory

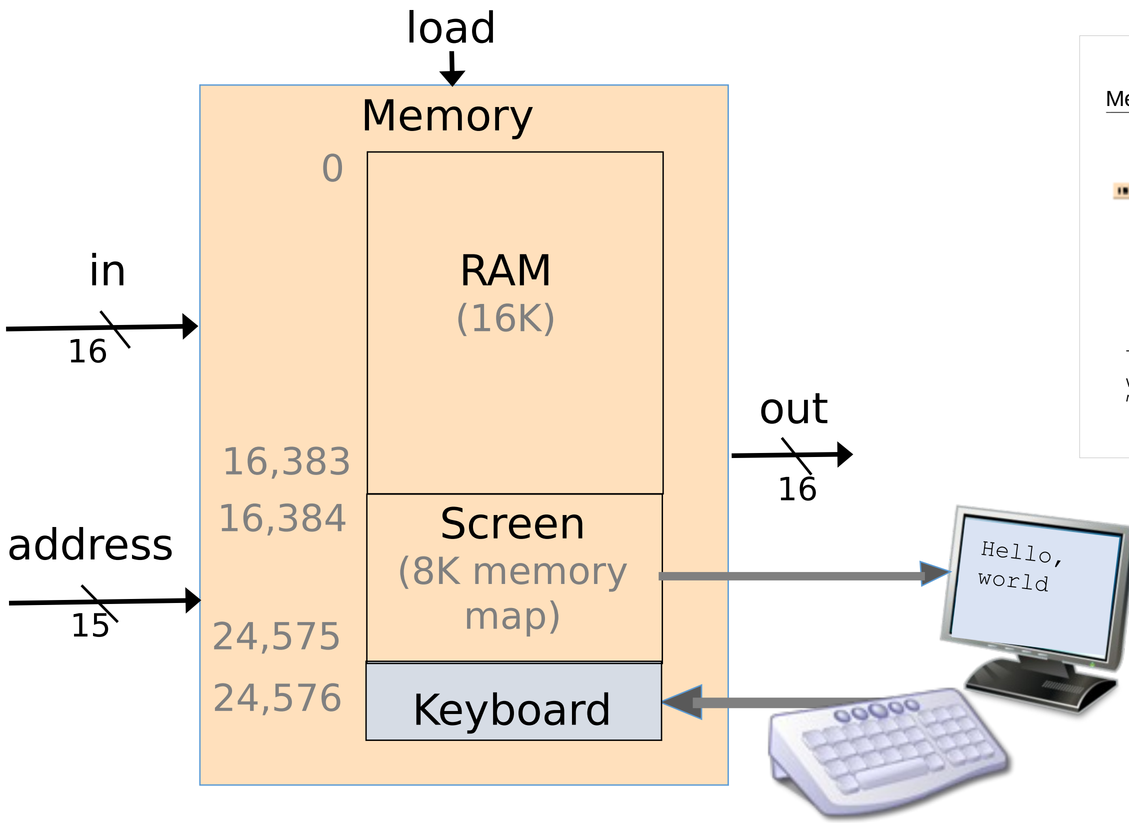


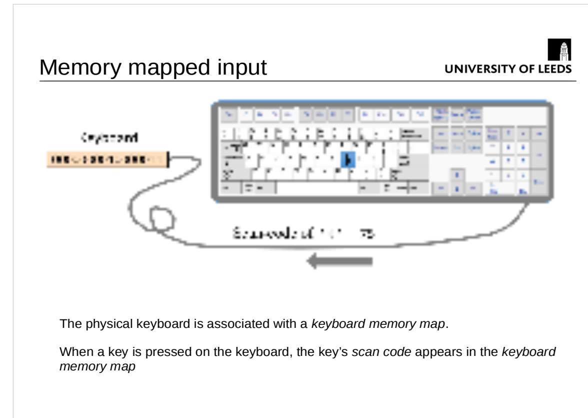




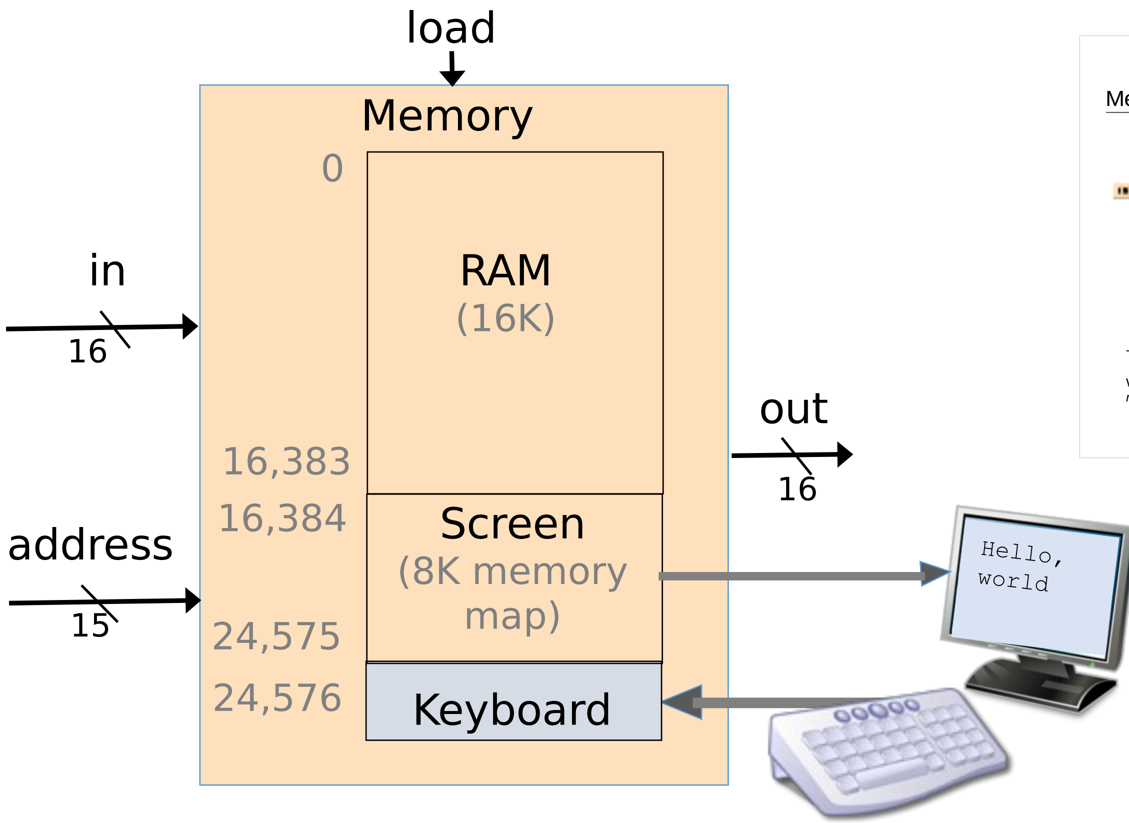
- The Hack screen is realized by a built-in chip named Screen
- Screen: a regular RAM + display output side-effect.

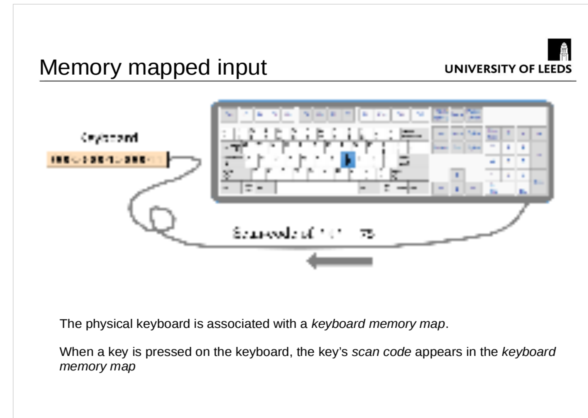
Hack Computer: Data Memory

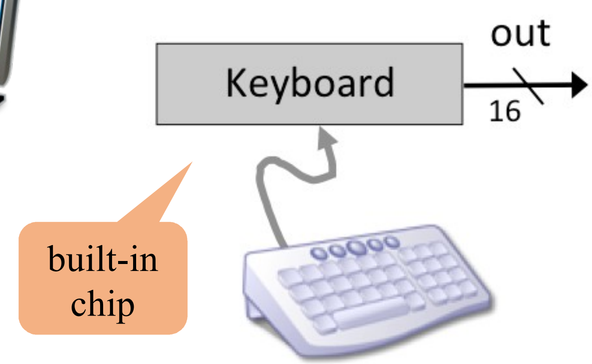




Hack Computer: Instruction Memory



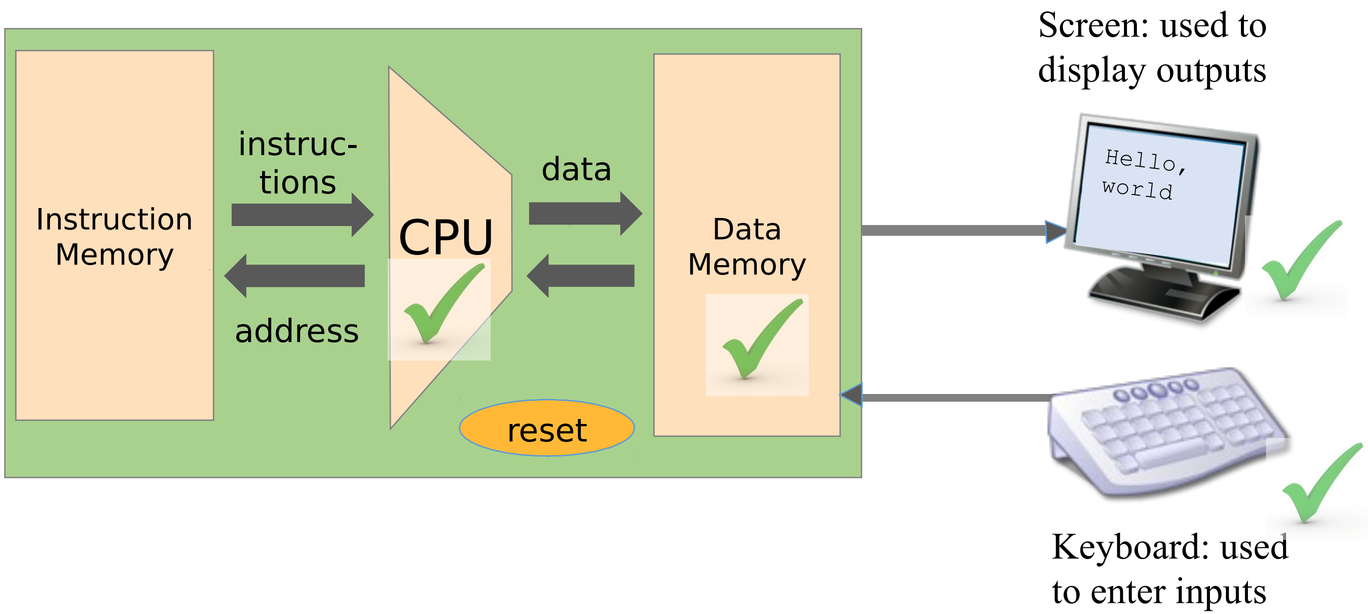




- Realized by a built-in chip named `Keyboard`
- `Keyboard`: A read-only 16-bit register + a keyboard input side-effect.

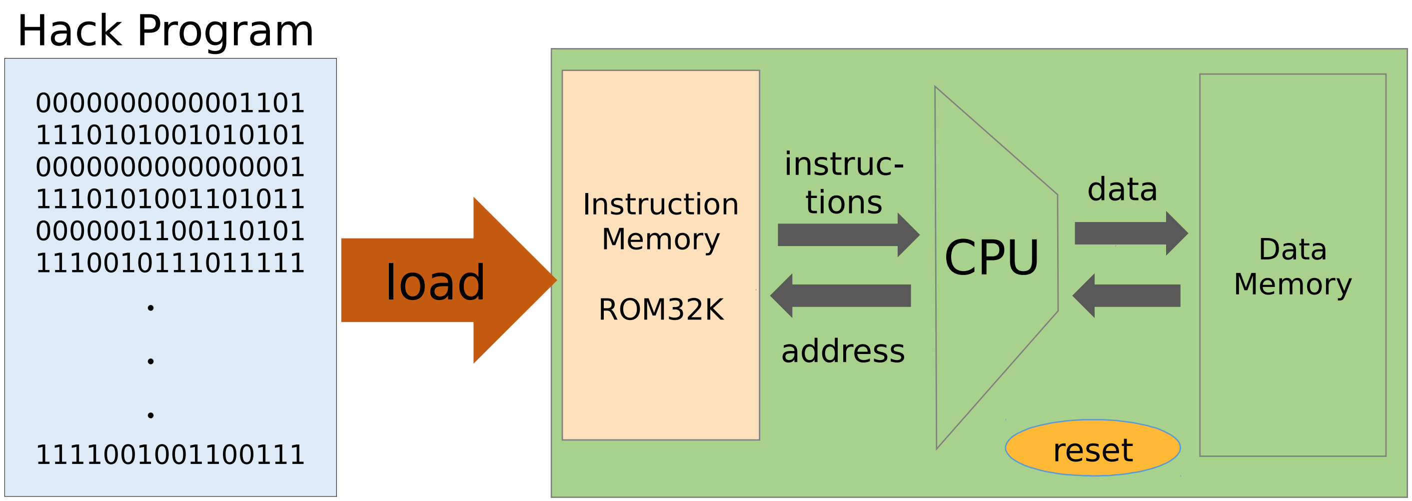
Hack Computer: Instruction Memory

UNIVERSITY OF LEEDS



Hack Computer: Instruction Memory

UNIVERSITY OF LEEDS



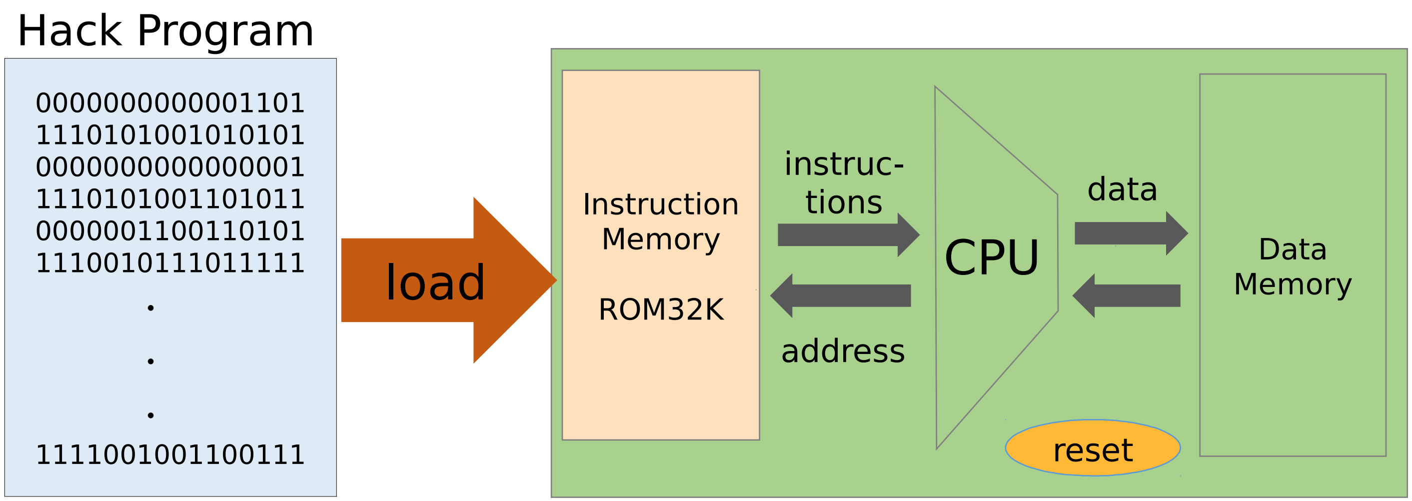
<u>To run a program on the Hack computer:</u>

- Load the program into the Instruction Memory
- Press “reset”
- The program starts running.

HOW?

Hack Computer: Instruction Memory

UNIVERSITY OF LEEDS

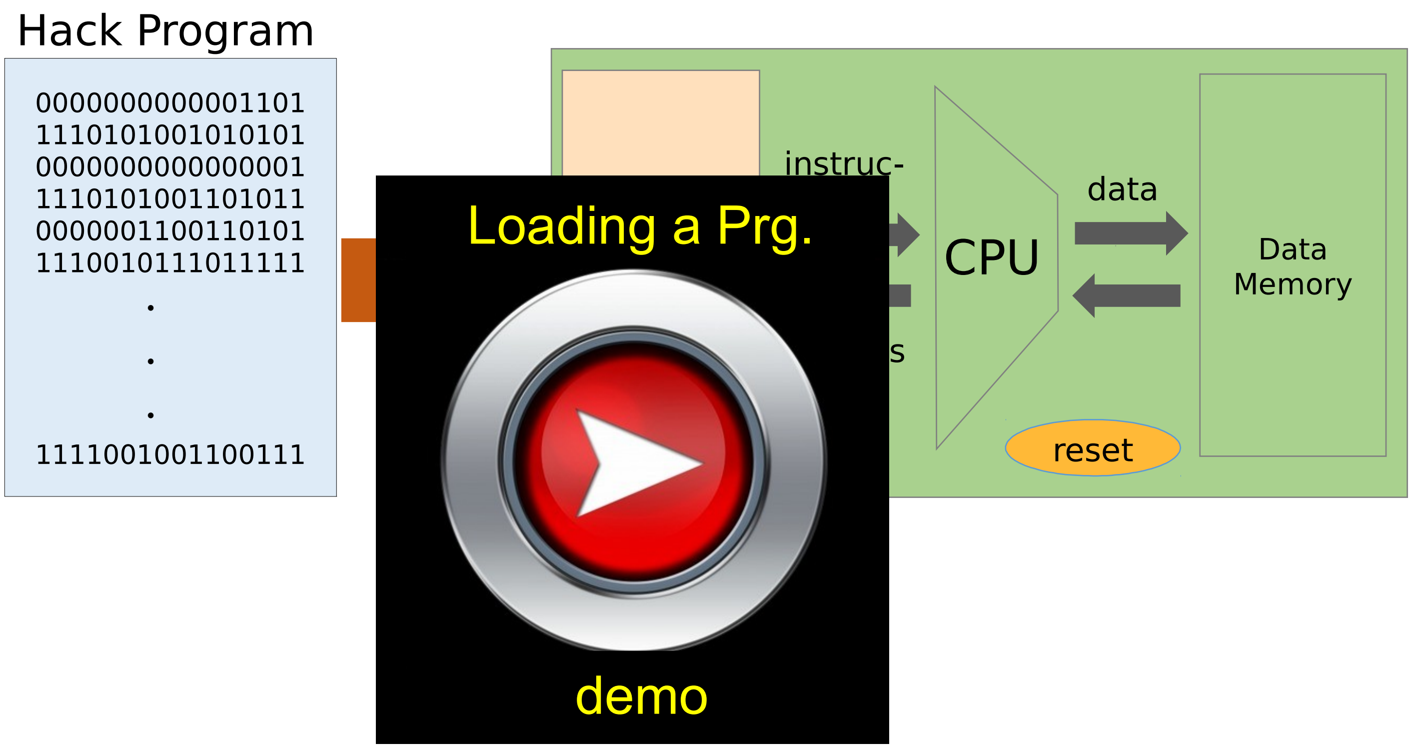


<u>Loading a program into the Instruction Memory:</u>

- <u>Hardware implementation:</u> plug-and-play ROM chips
  (each comes pre-loaded with a program's code)

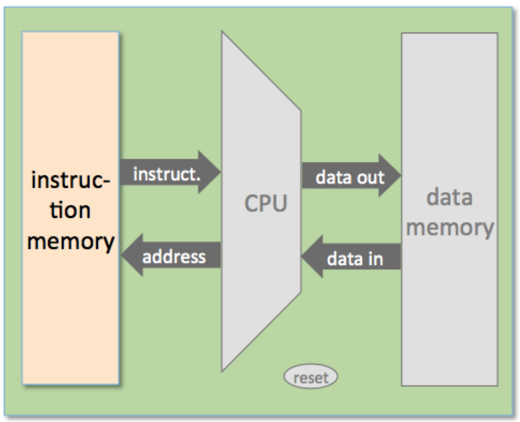
- <u>Hardware simulation:</u> programs are stored in text files;
  (the simulator's software features a load-program service)

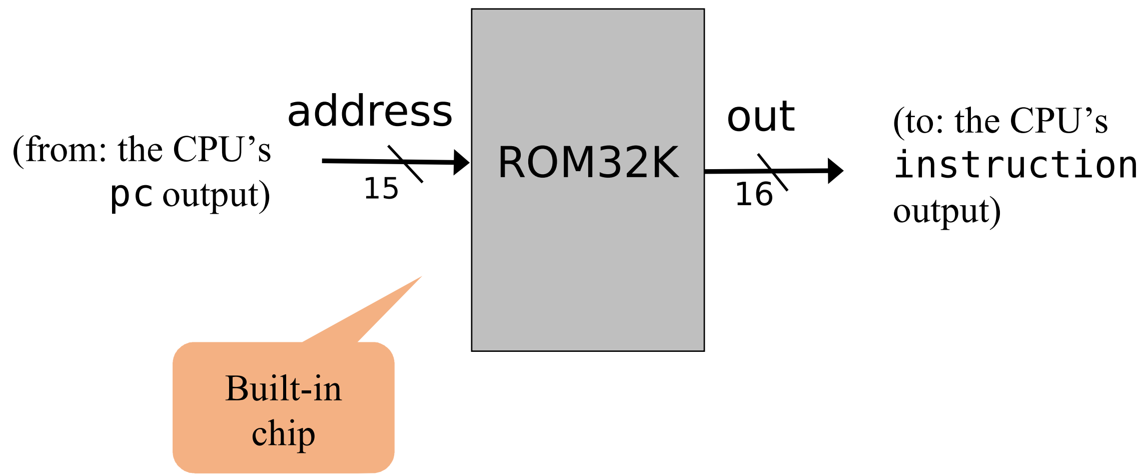
Hack Computer: Instruction Memory



Tutorial Video from
Nand2Tetris Course

Hack Computer: Instruction Memory

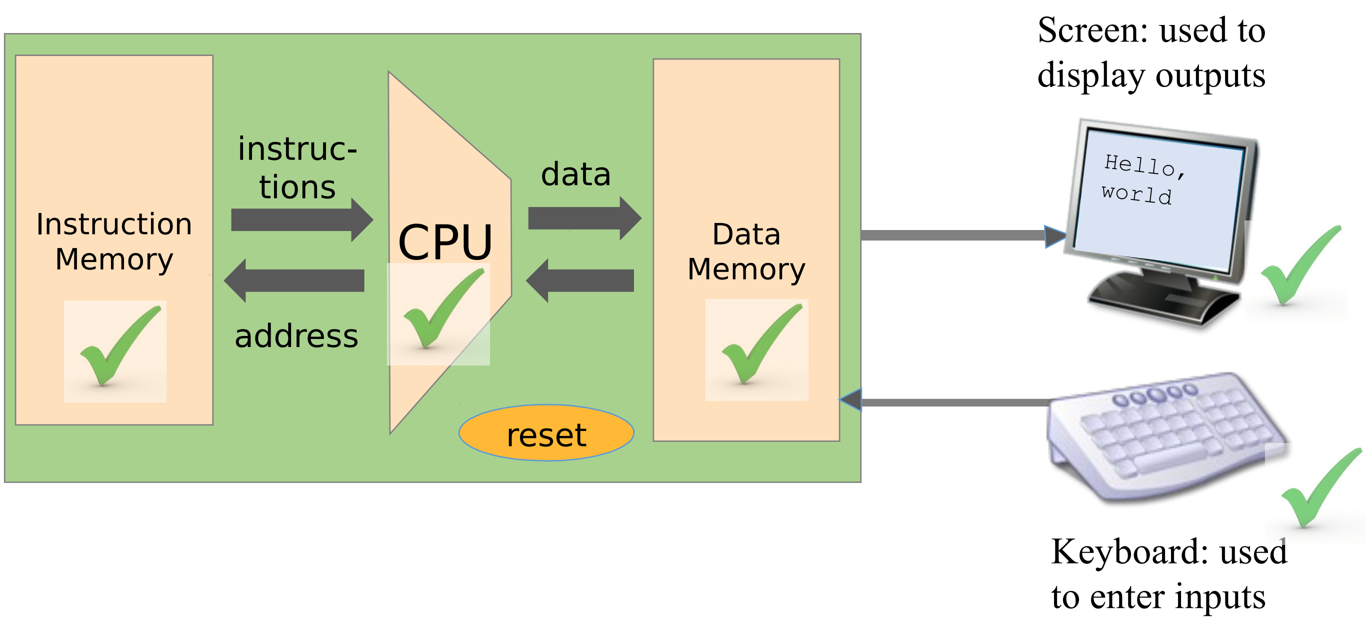




- The Hack Instruction Memory is realized by a built-in chip named `ROM32K`
- `ROM32K`: a read-only, 16-bit, 32K RAM chip + program loading side-effect.

Hack Computer: Implementation

UNIVERSITY OF LEEDS



Hack Computer: Implementation

UNIVERSITY OF LEEDS