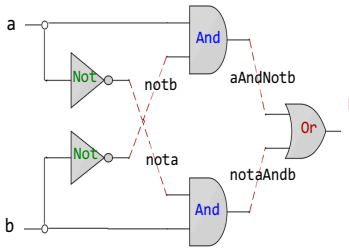
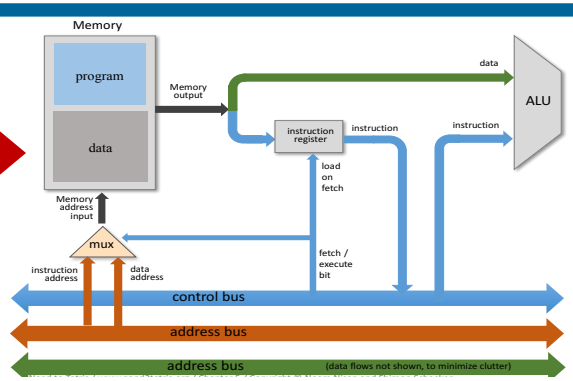
Computer Organization & Assembly Language Programming




```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```




```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

Lecture # 22

Data Path of Hack CPU - I

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

Slides of first half of the course are adapted from:
https://www.nand2tetris.org
Download s/w tools required for first half of the course from the following link:
https://drive.google.com/file/d/0B9c0BdDJz6XpZUh3X2dPR1o0MUE/view

Instructor: Muhammad Arif Butt, Ph.D.

Today's Agenda

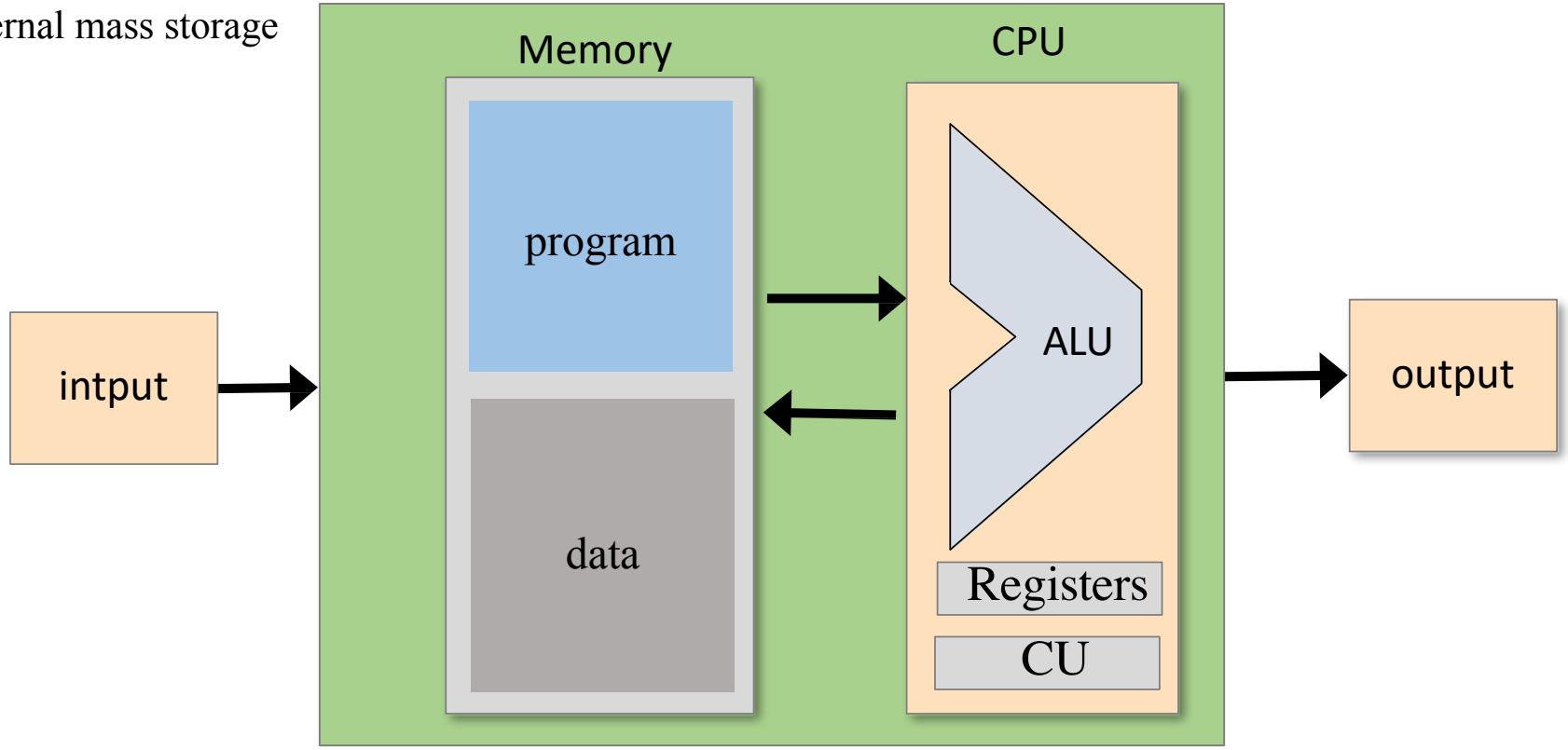
- Von Neumann Architecture
- Flow of Information inside Computers
- Buses
  - Data Bus
  - Address Bus
  - Control Bus
- Fetch Execute Cycle
- Fetch Execute Clash
- Harvard Architecture

Von Neumann Architecture

Von Neumann Architecture

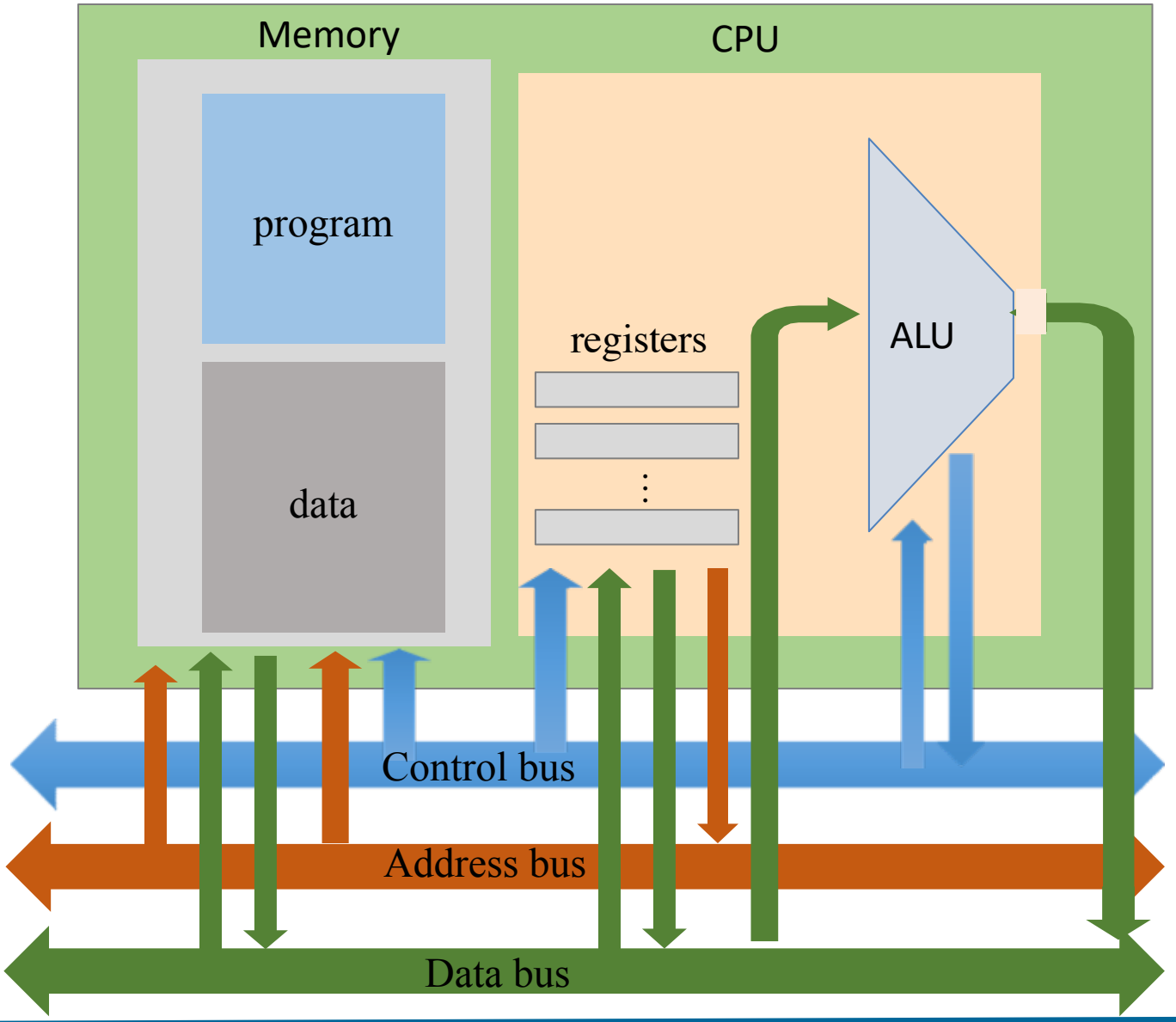
The Von Neumann architecture is a computer architecture given by a mathematician and physicist John von Neumann describes the design architecture for an electronic digital computer with these components:

- A Processing Unit that contains an ALU and registers
- A Control Unit that contains an instruction register and program counter
- A Memory unit that stores data and instructions
- An Input and Output mechanism
- An external mass storage

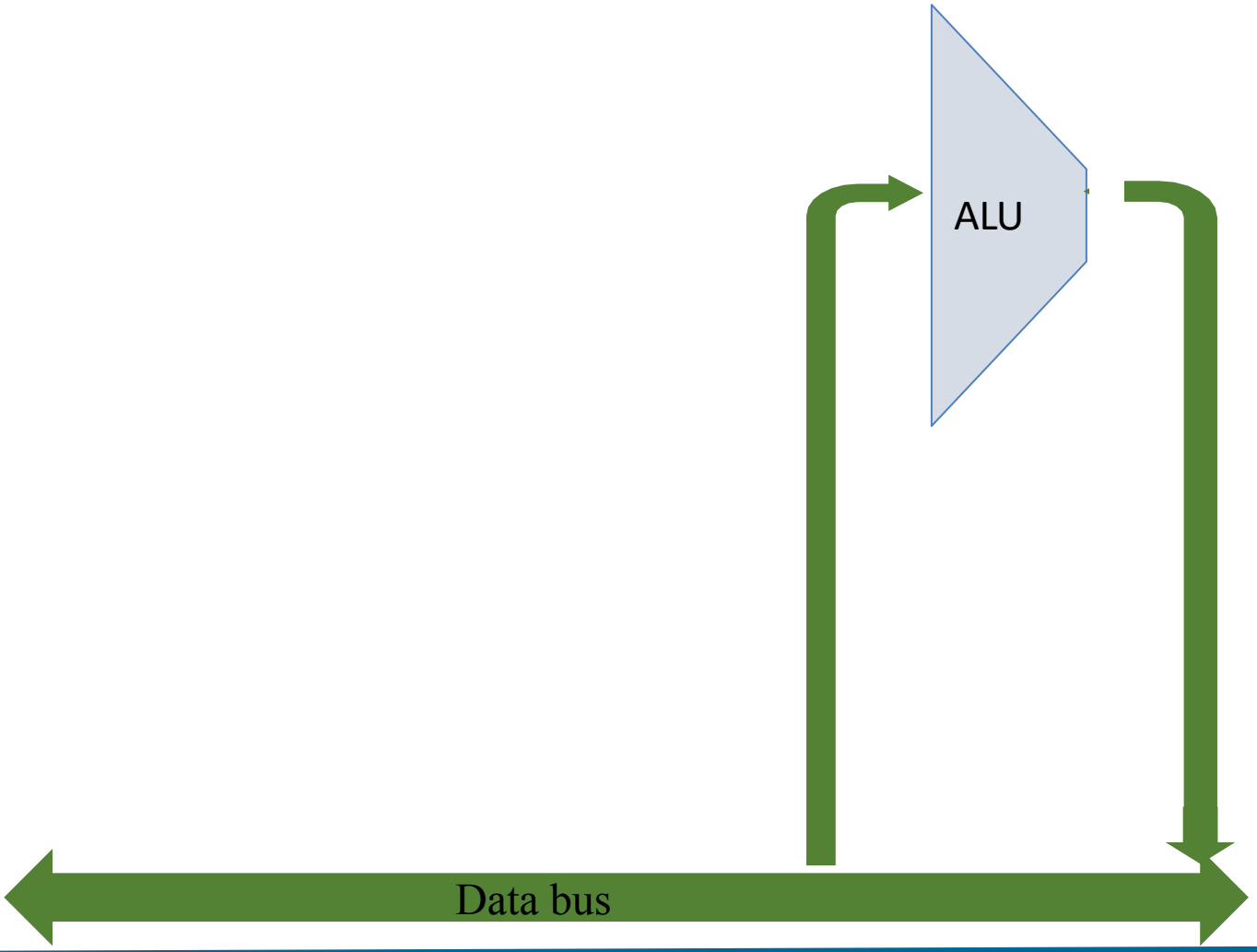


Flow of Information inside a Computer System

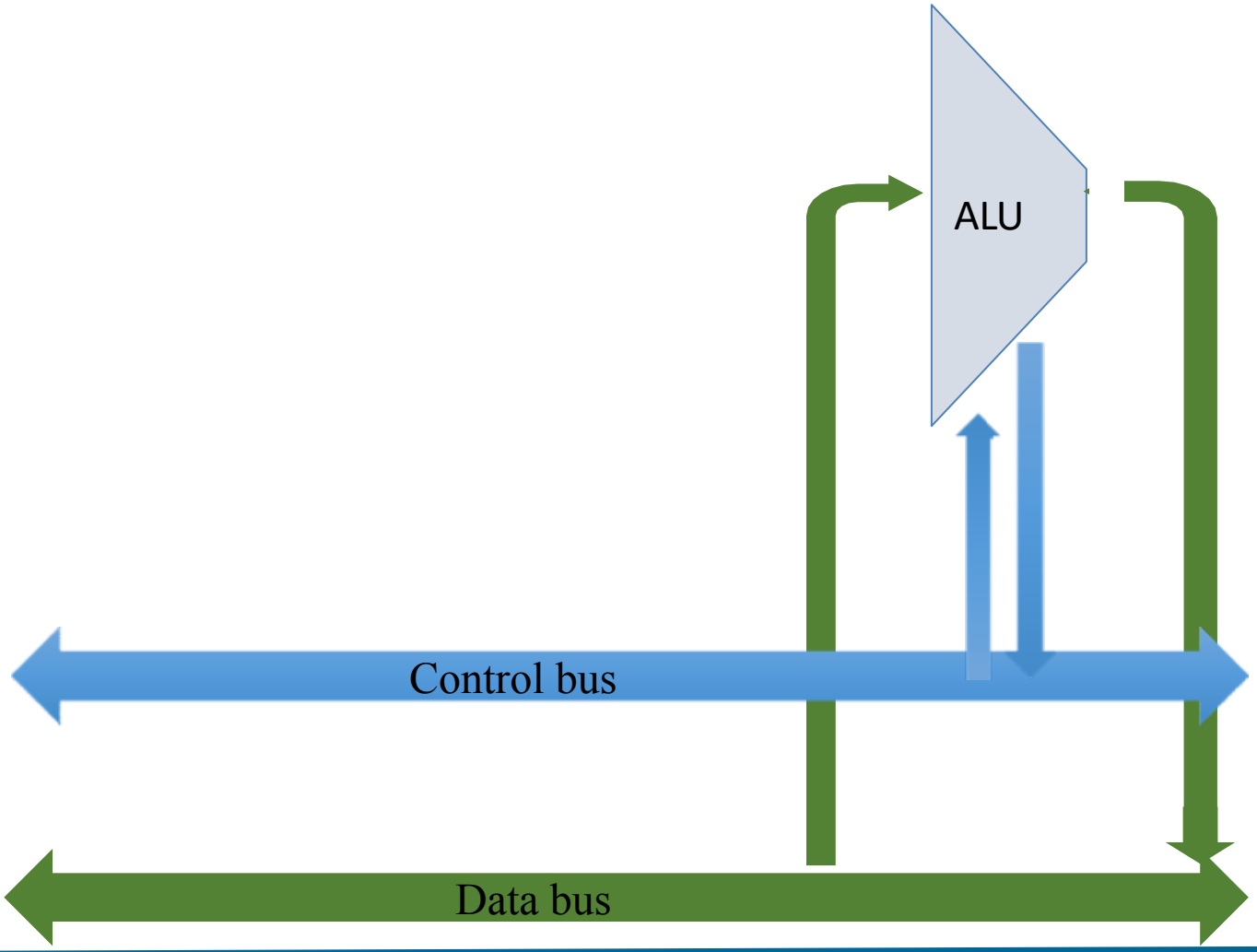
Information Flow



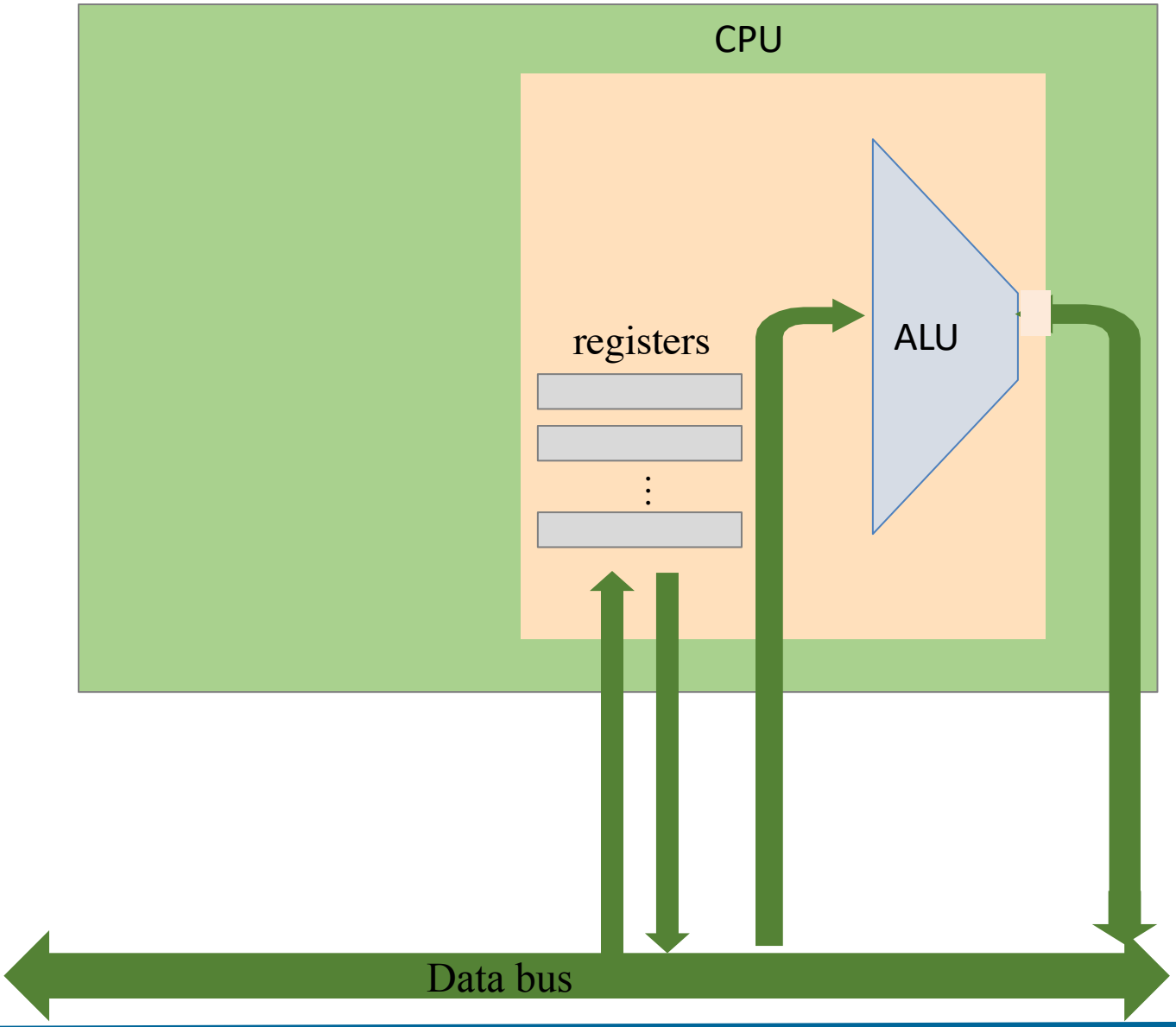
The Arithmetic Logic Unit



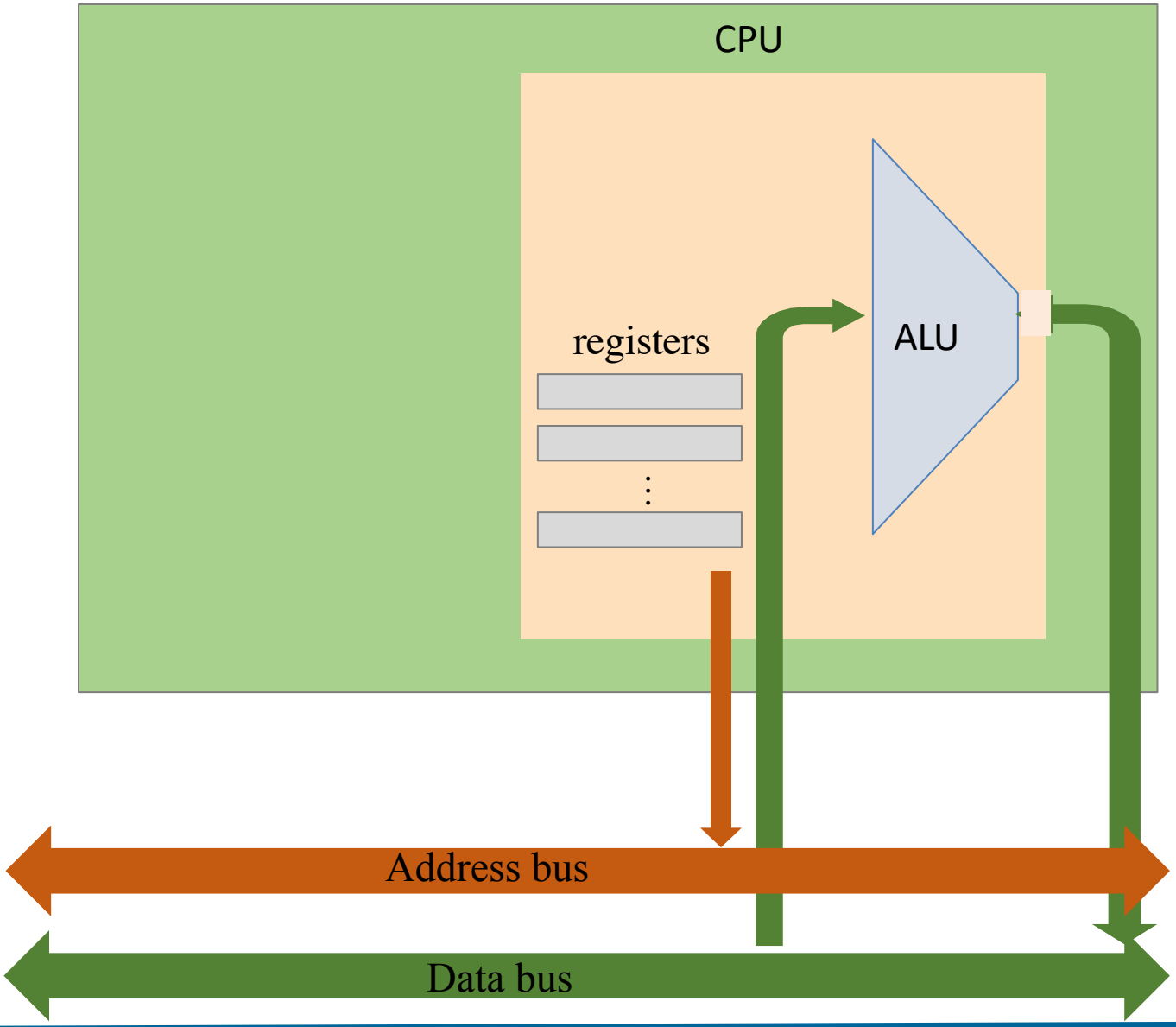
The Control



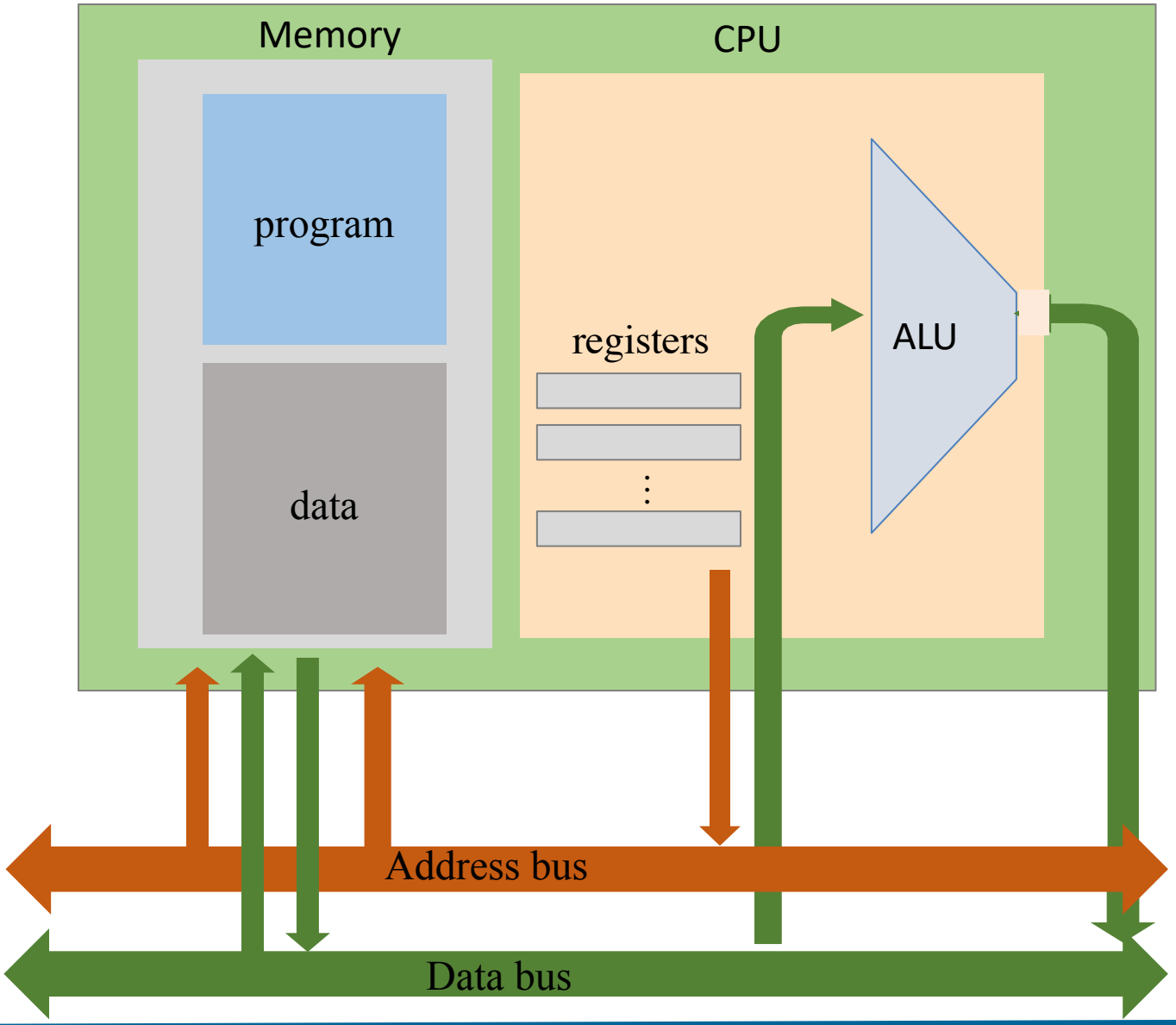
Data Registers



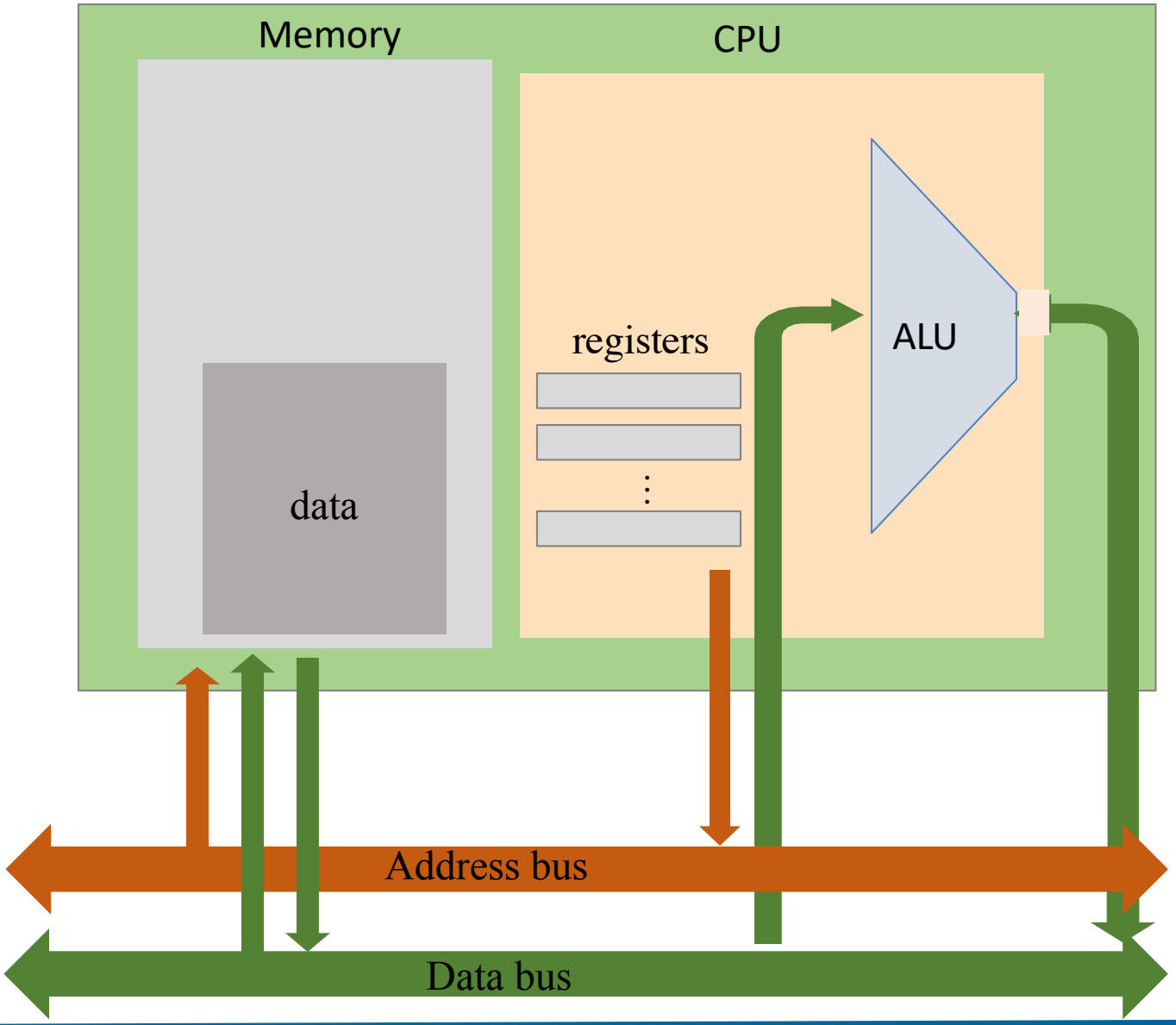
Address Registers



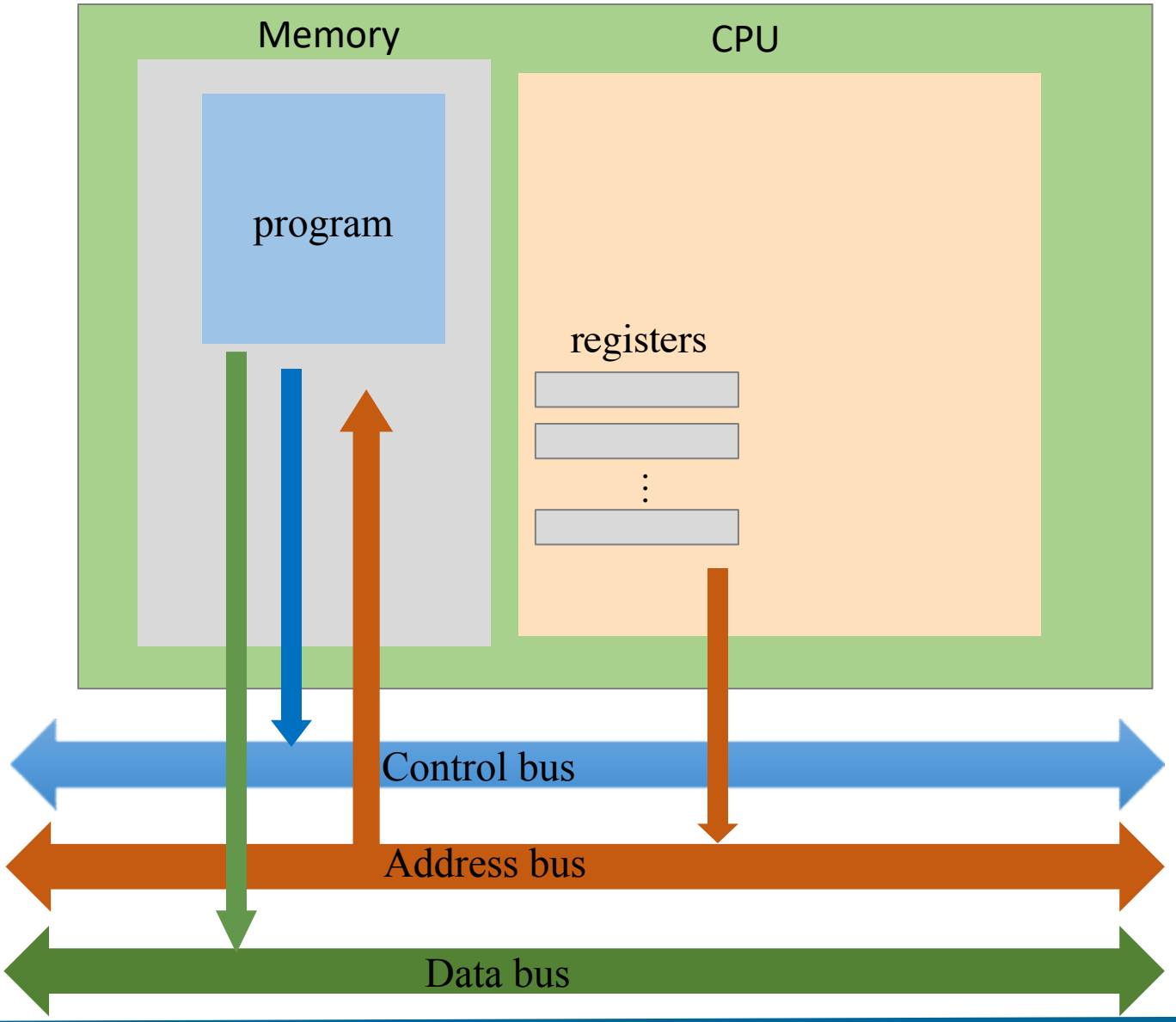
Memory



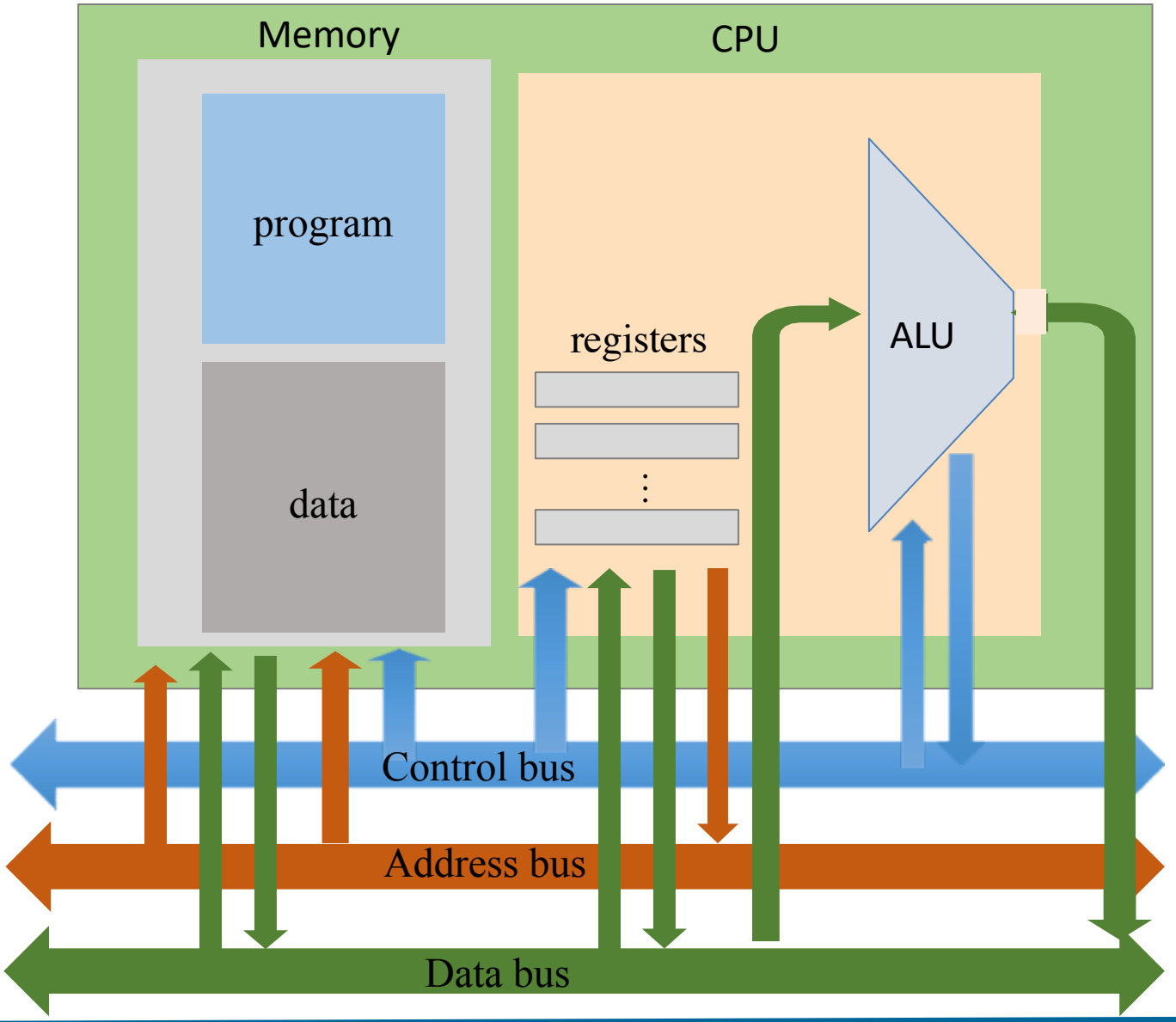
Data Memory



Program Memory



Overall Picture



Overview of General Fetch-Execute Cycle

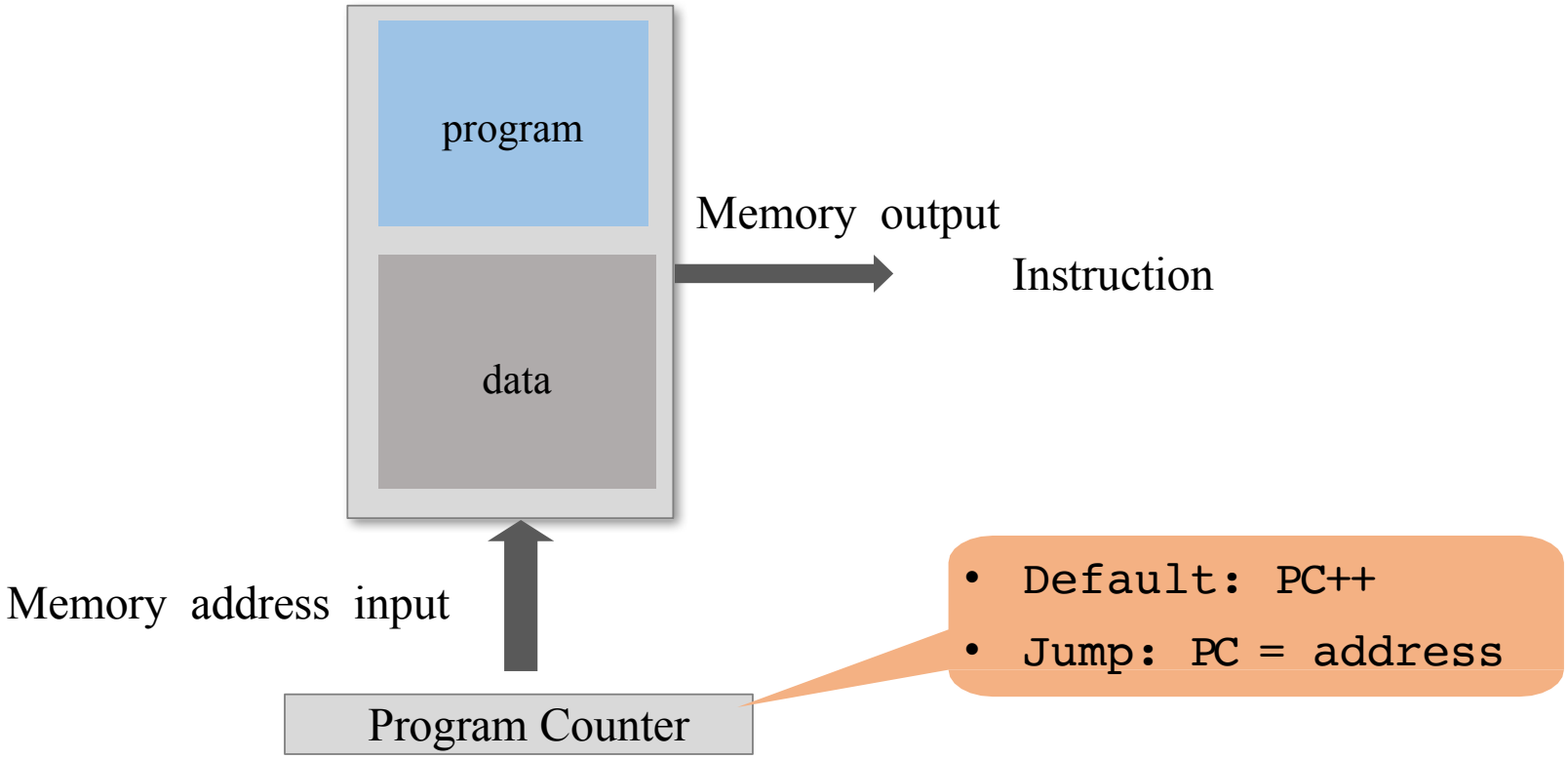
Basic CPU Loop

Repeat:

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

Fetching

- Put the location of the next instruction in the Memory address input
- Read the contents of the memory from that location to get the instruction code



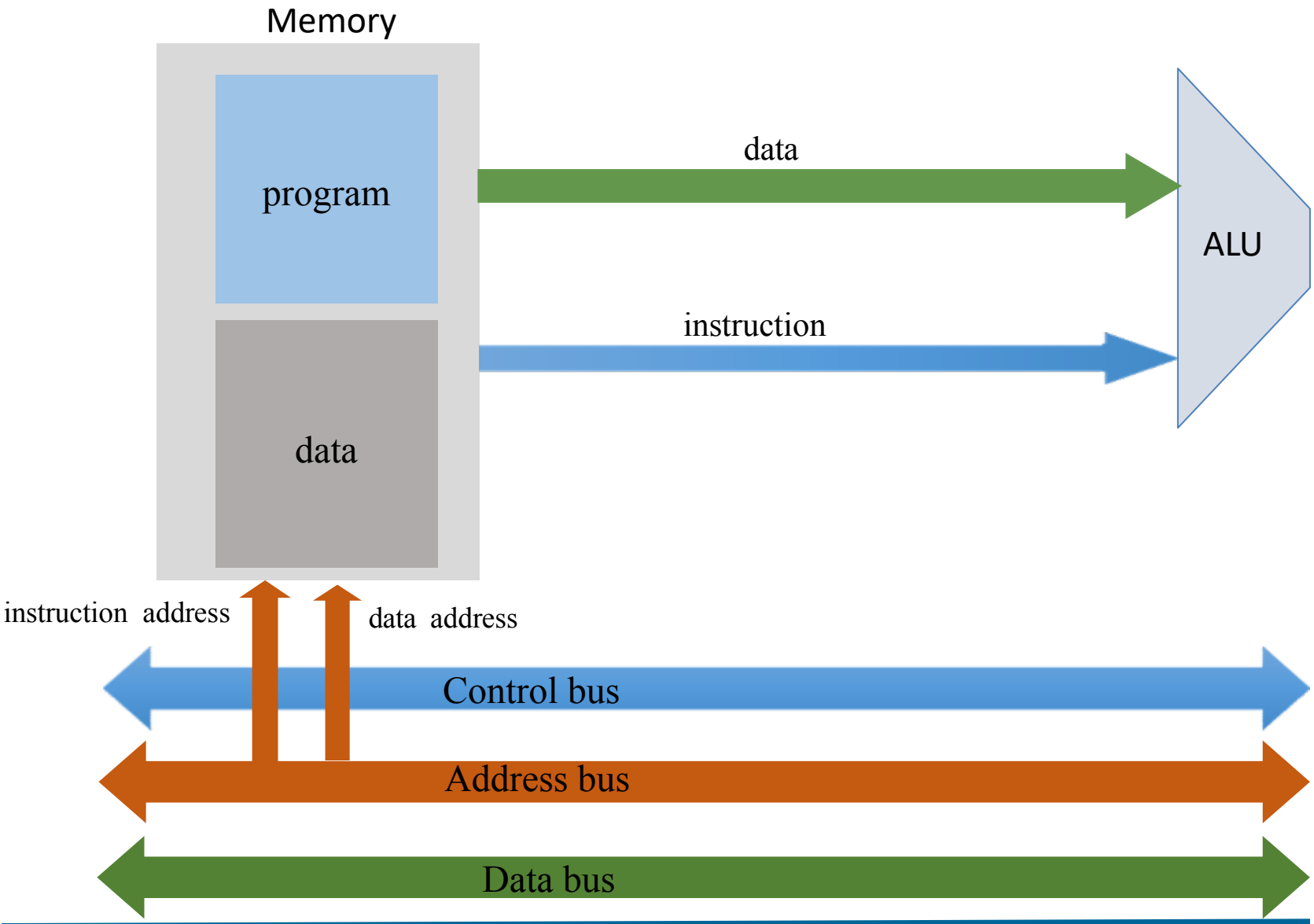
Executing

- 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
  - …

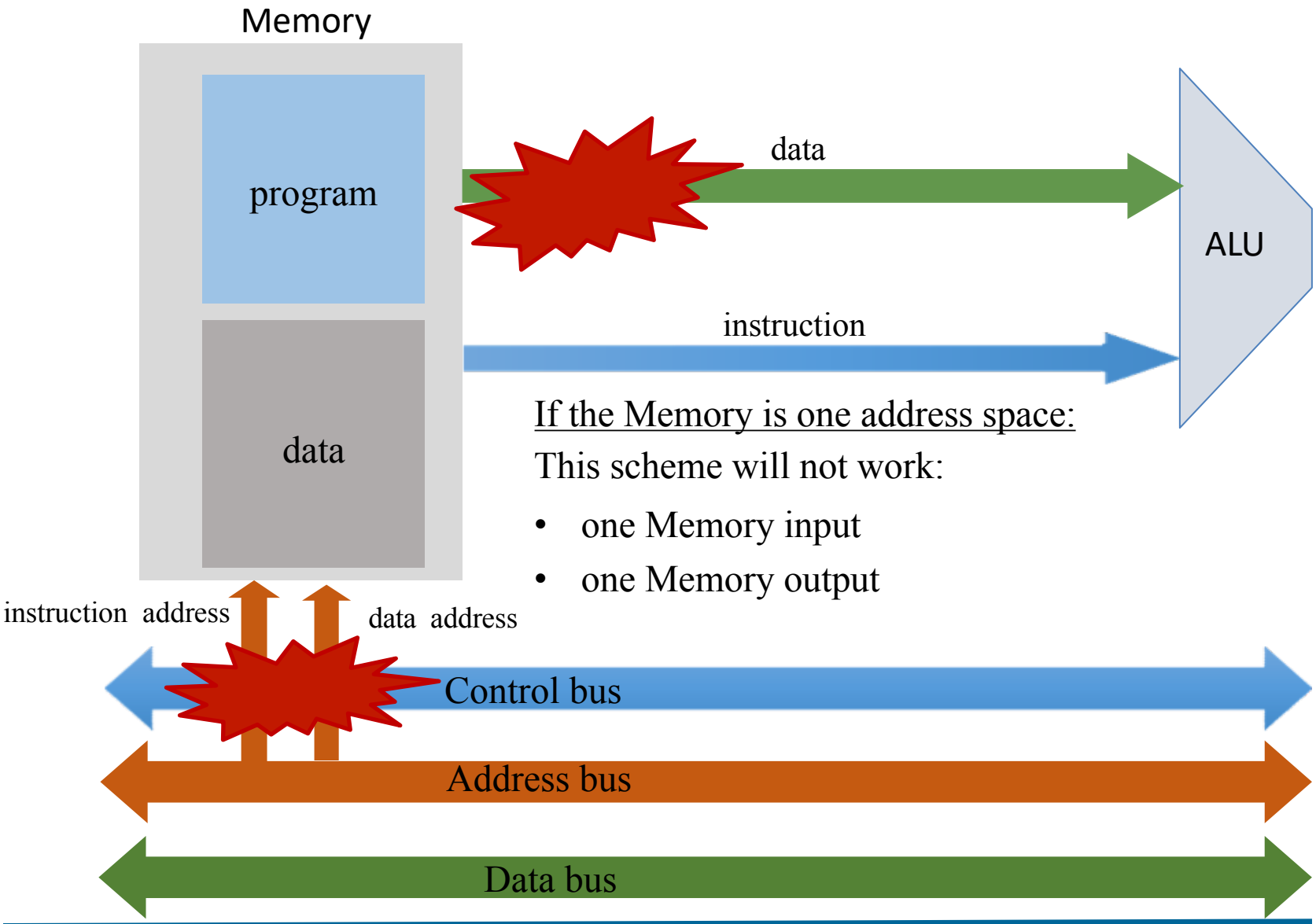
Different subset of the instruction bits controls different aspects of the operation

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

Fetch Execute



Fetch-Execute Clash

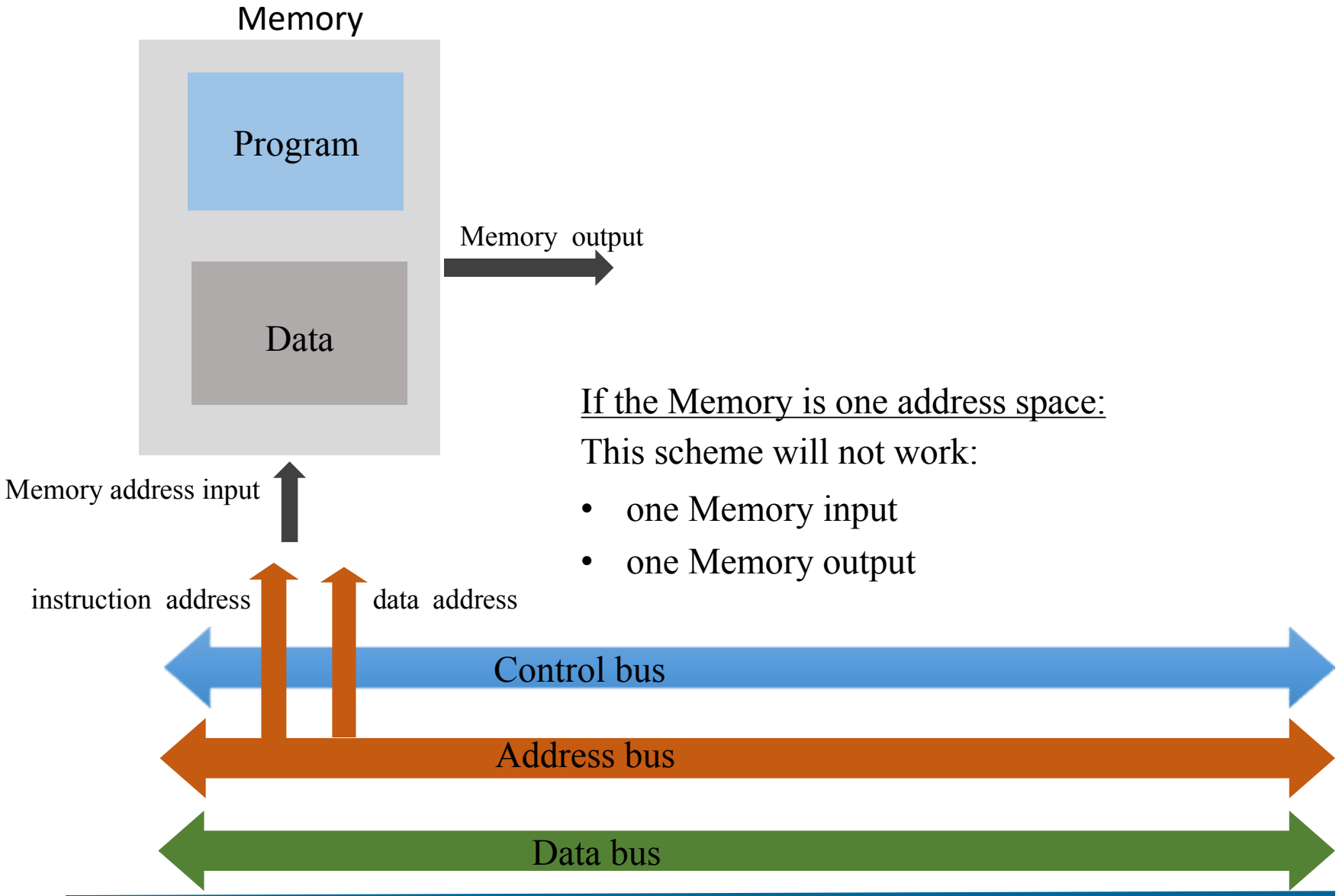


If the Memory is one address space:

This scheme will not work:

- one Memory input
- one Memory output

Fetch-Execute Clash (cont…)

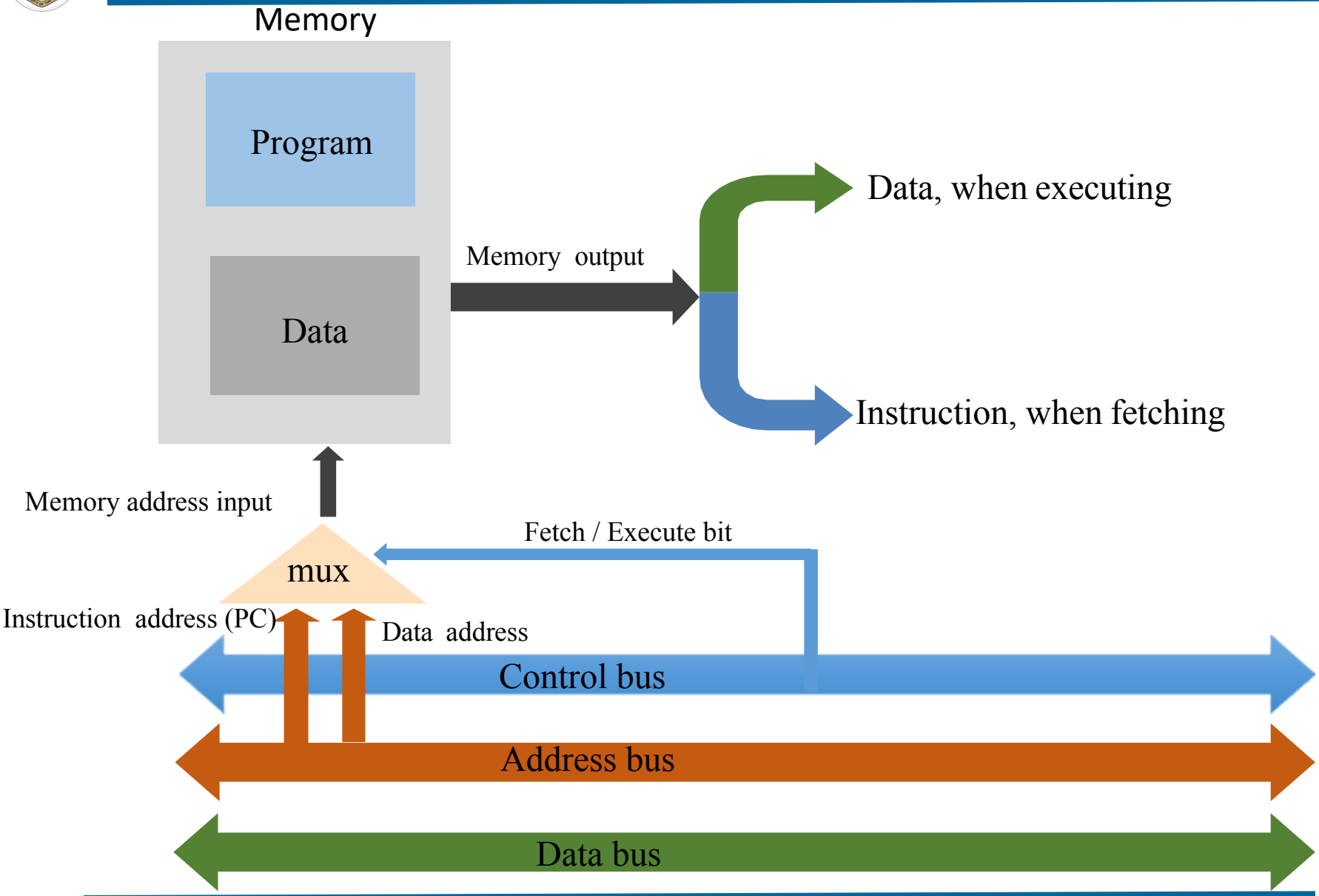


If the Memory is one address space:

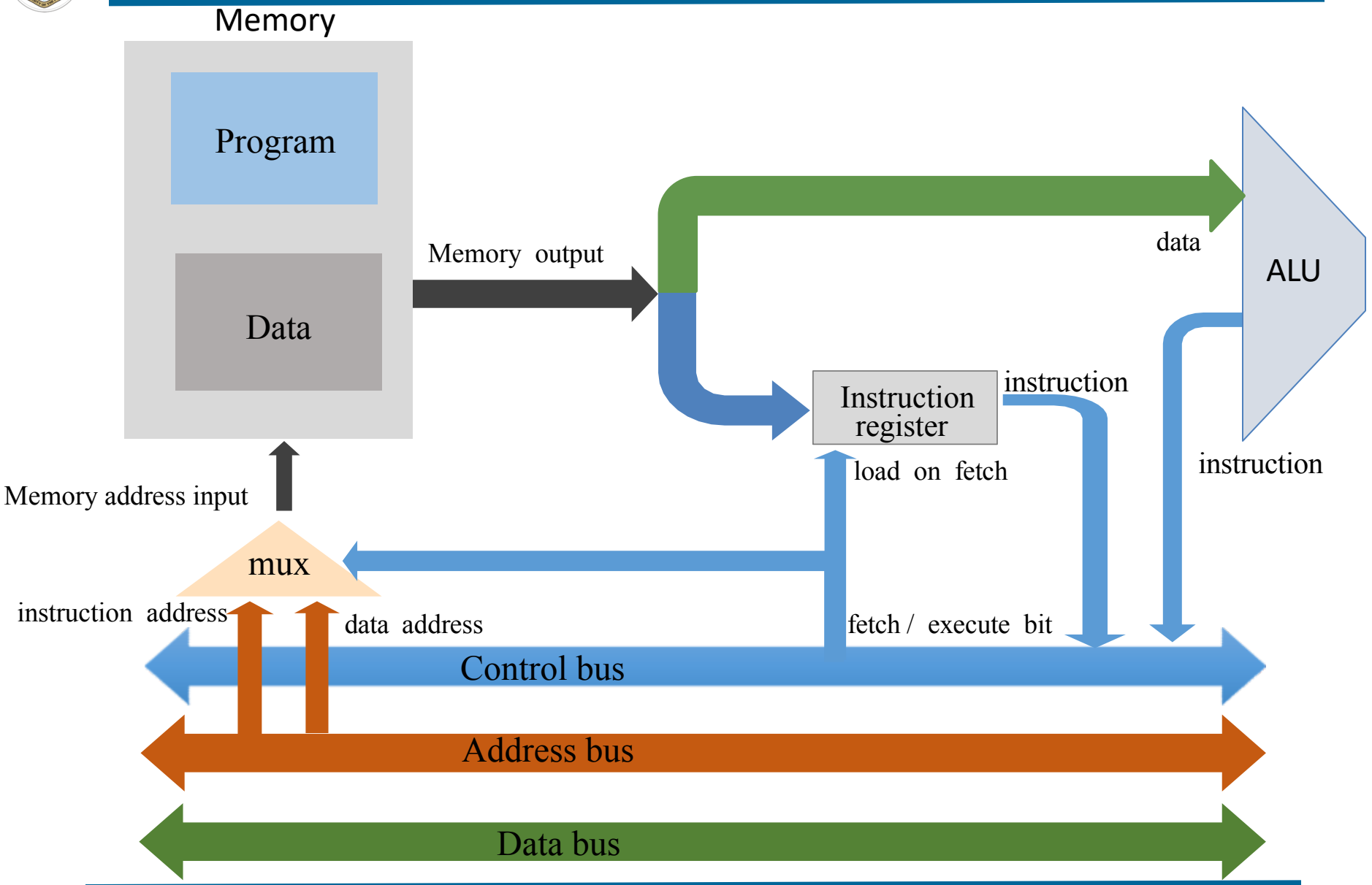
This scheme will not work:

- one Memory input
- one Memory output

Solution: multiplex, using an instruction register



Solution: multiplex, using an instruction register



Simpler Solution: Harvard Architecture

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

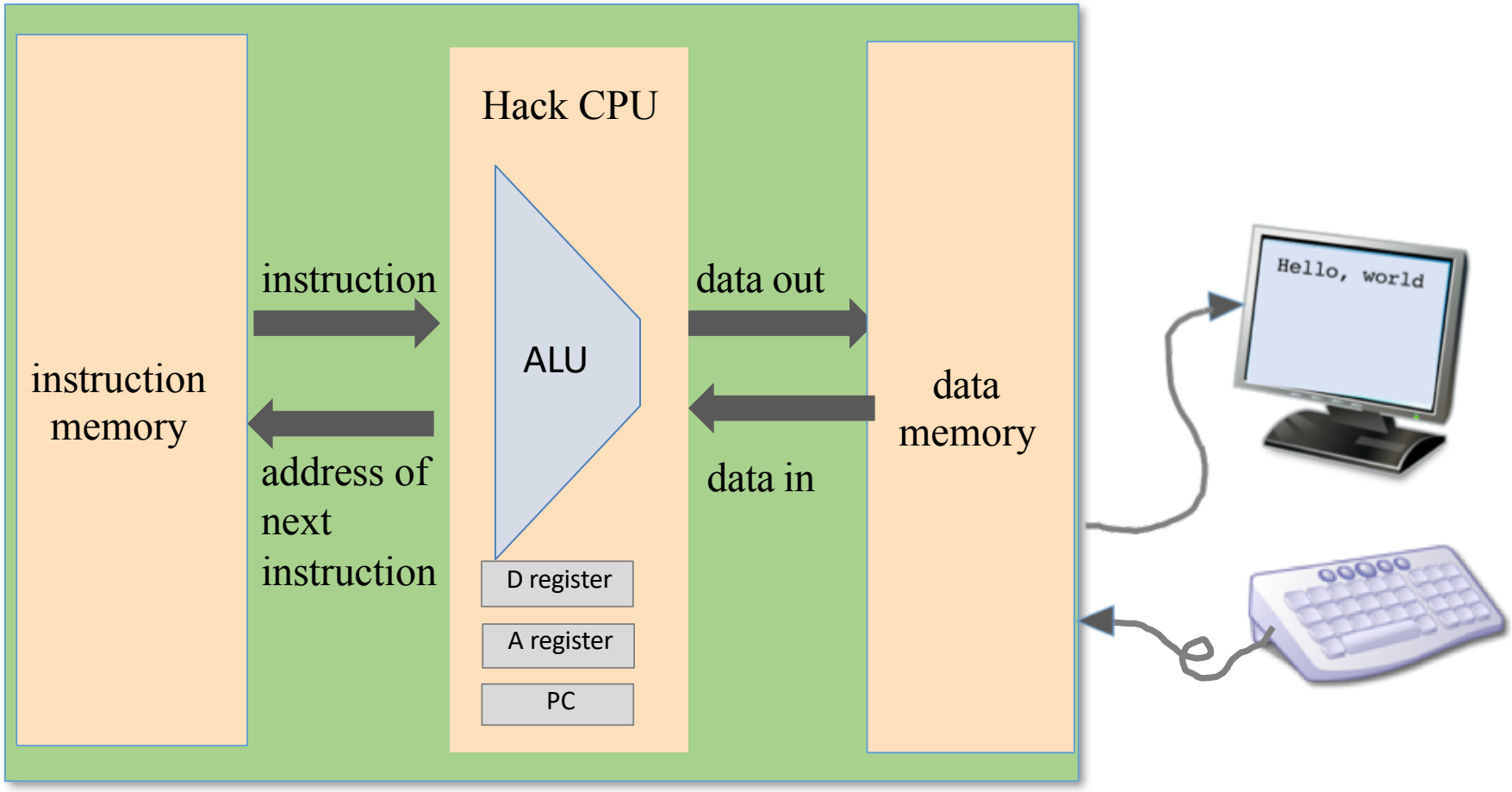
Advantage:

- Complication avoided

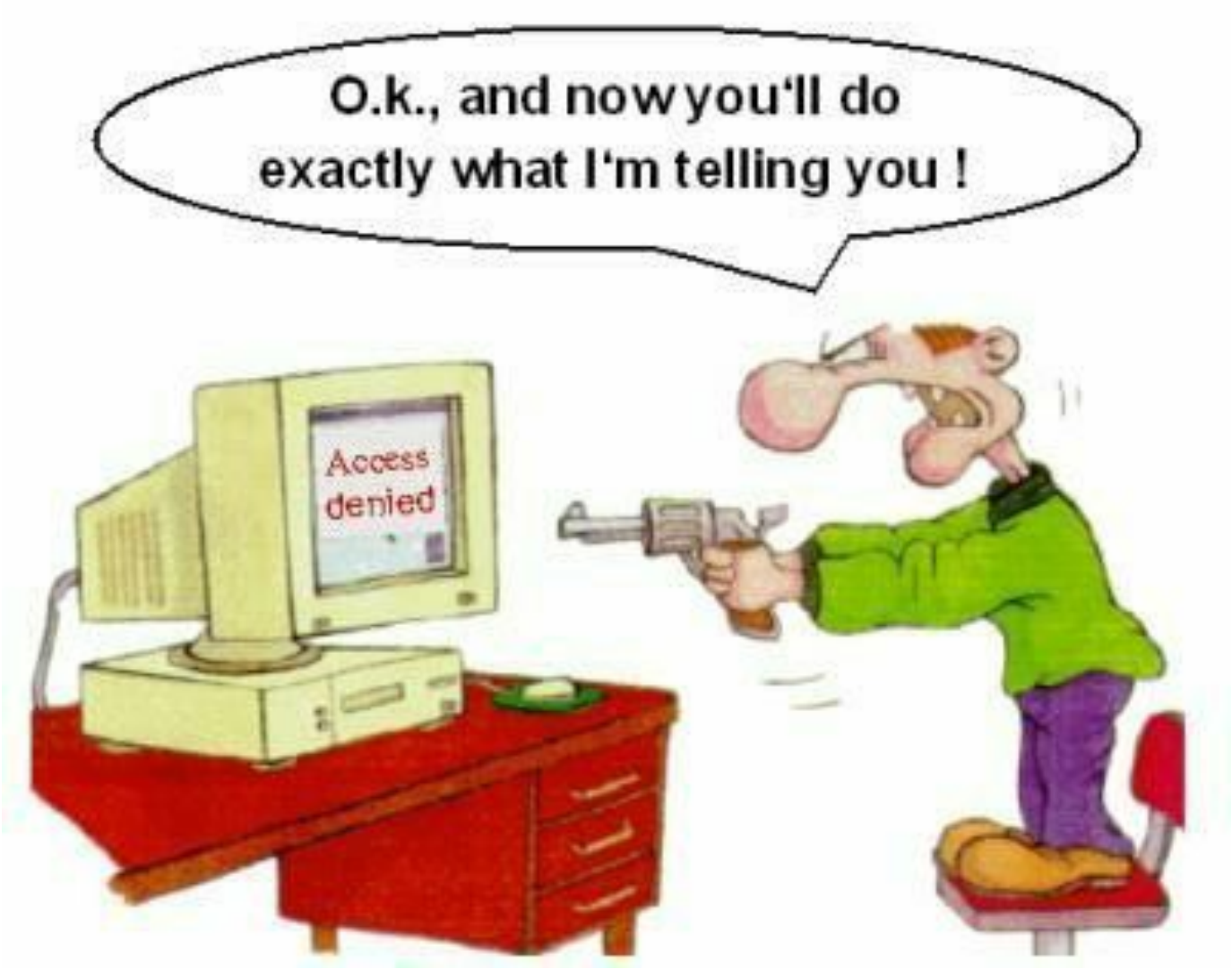
Disadvantage:

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

Hack Computer Architecture



Things To Do


O.k., and now you'll do exactly what I'm telling you !
Access denied