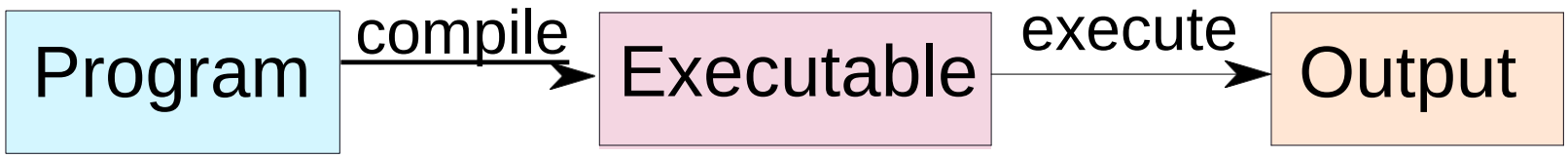
CSE112: Computer Organization (Section A)

Instructor: Sujay Deb

Lecture 4

How to Instruct a Computer?



- Write a program in a high level language – C, C++, Java
- **Compile** it into a format that the computer understands
- Execute the program

Instruction Set Architecture:

The semantics of all the instructions supported by a processor is known as its instruction set architecture (ISA). This includes the semantics of the instructions themselves, along with their operands, and interfaces with peripheral devices.

Features of an ISA

- Example of instructions in an ISA
  - Arithmetic instructions : add, sub, mul, div
  - Logical instructions : and, or, not
  - Data transfer/movement instructions
- Complete
  - It should be able to implement all the programs that users may write.

Features of an ISA – II

- Concise
  - The instruction set should have a limited size. Typically an ISA contains 32-1000 instructions.
- Generic
  - Instructions should not be too specialized, e.g. add14 (adds a number with 14) instruction is too specialized
- Simple
  - Should not be very complicated.

Designing an ISA

- Important questions that need to be answered :
  - How many instructions should we have ?
  - What should they do ?
  - How complicated should they be ?

Two different paradigms : RISC and CISC

RISC
(Reduced Instruction Set Computer)

CISC
(Complex Instruction Set Computer)

RISC vs CISC

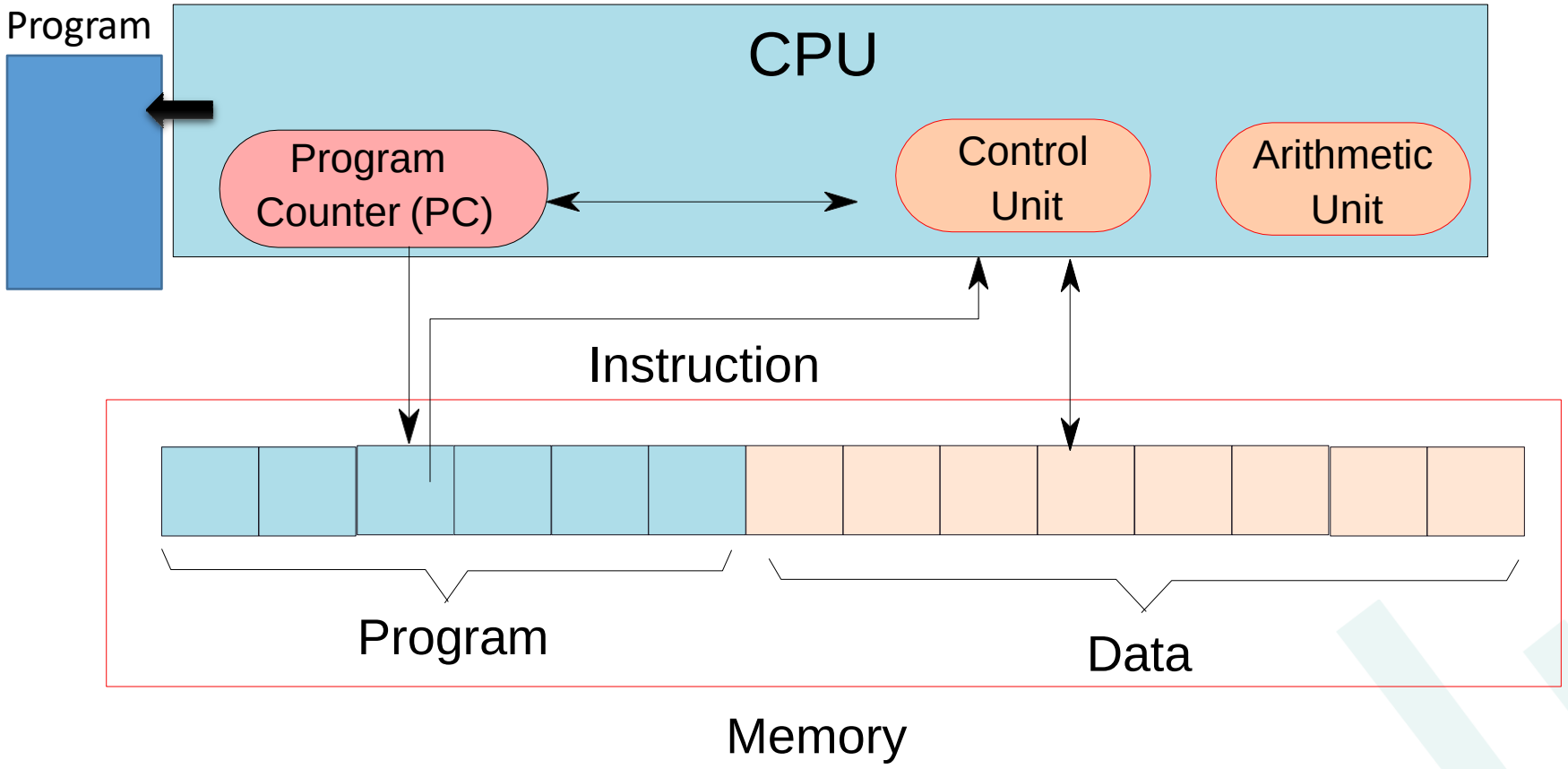
A reduced instruction set computer (RISC) implements simple instructions that have a simple and regular structure. The number of instructions is typically a small number (64 to 128). Examples: ARM, IBM PowerPC, HP PA-RISC

A complex instruction set computer (CISC) implements complex instructions that are highly irregular, take multiple operands, and implement complex functionalities. Secondly, the number of instructions is large (typically 500+). Examples: Intel x86, VAX

Summary so far …

- Computers are dumb yet ultra-fast machines.
- Instructions are basic rudimentary commands used to communicate with the processor. A computer can execute billions of instructions per second.
- The compiler transforms a user program written in a high level language such as C to a program consisting of basic machine instructions.
- The instruction set architecture(ISA) refers to the semantics of all the instructions supported by a processor.
- The instruction set needs to be complete. It is desirable if it is also concise, generic, and simple.

Compute Block Diagram



Elements of a Computer

- Memory (array of bytes) contains
  - The program, which is a sequence of instructions
  - The program data → variables, and constants
- The program counter(PC) points to an instruction in a program
  - After executing an instruction, it points to the next instruction by default
  - A branch instruction makes the PC point to another instruction (not in sequence)
- CPU (Central Processing Unit) contains the
  - Program counter, instruction execution units

Let us now design an ISA …

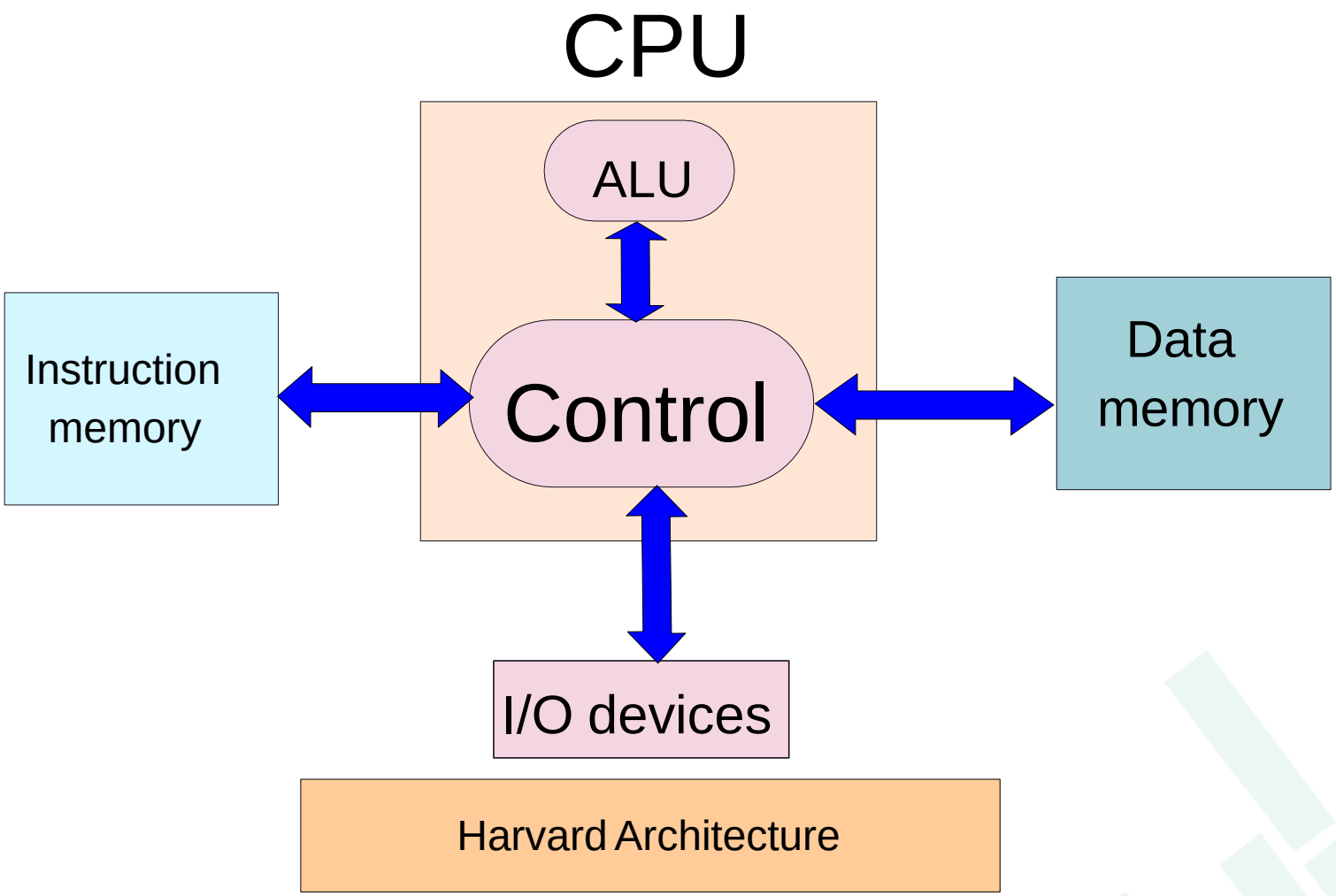
- Single Instruction ISA
  - sbn – subtract and branch if negative
- Add (a + b) (assume temp = 0)

```
1: sbn temp, b, 2
2: sbn a, temp, exit
```

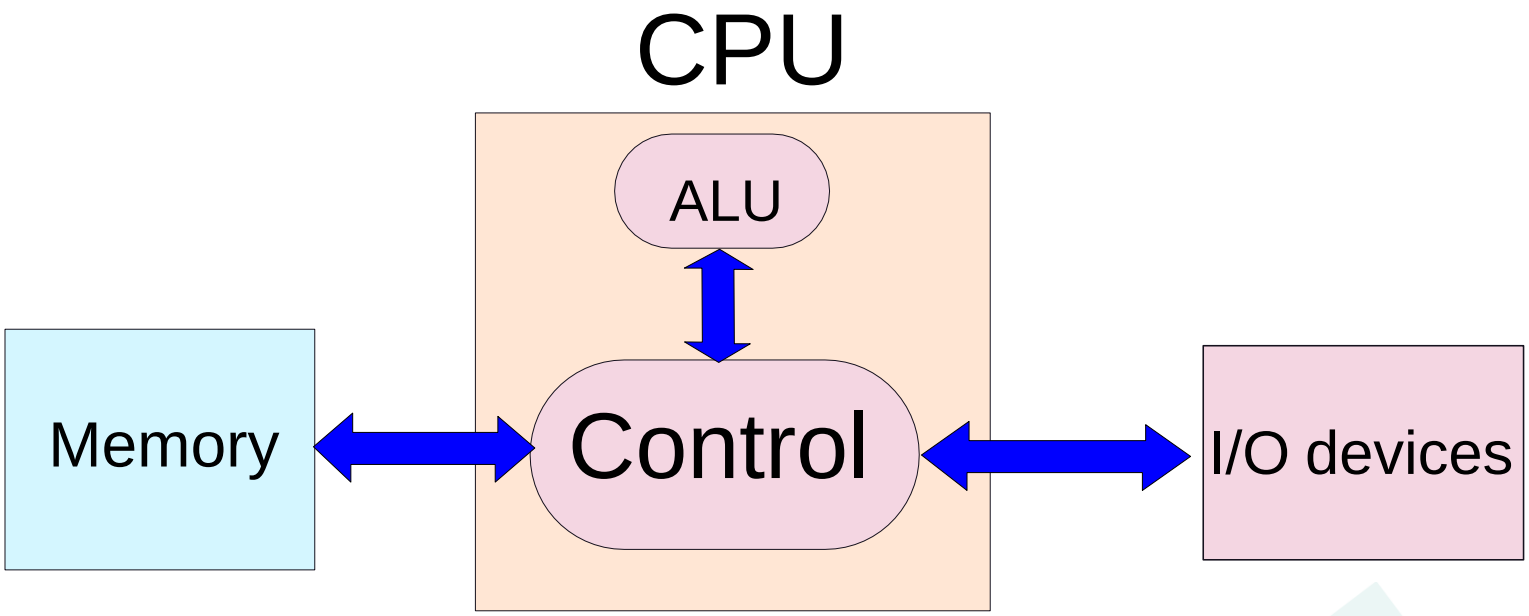
Multiple Instruction ISA

- Arithmetic Instructions
  - add, subtract, multiply, divide
- Logical Instructions
  - or, and, not
- Move instructions
  - Transfer values between memory locations
- Branch instructions
  - Move to a new program location, based on the values of some memory locations

Designing Practical Machines



Von-Neumann Architecture



Problems with Harvard/ Von-Neumann Architectures

- The memory is assumed to be one large array of bytes
  - It is very very slow

General Rule: Larger is a structure, slower it is

- Solution:
  - Have a small array of named locations **(registers)** that can be used by instructions
  - This small array is very fast

Insight: Accesses exhibit locality (tend to use the same variables frequently in the same window of time)

Uses of Registers

- A CPU (Processor) contains set of registers (16-64)
- These are named storage locations.
- Typically values are loaded from memory to registers.
- Arithmetic/logical instructions use registers as input operands
- Finally, data is stored back into their memory locations.

Example of a Program in Machine Language with Registers

```
1: r1 = mem[b] // load b
2: r2 = mem[c] // load c
3: r3 = r1 + r2 // add b and c
4: mem[a] = r3 // save the result
```

- r1, r2, and r3, are registers
- mem → array of bytes representing memory

Machine with Registers