COMS12200
Introduction to
Computer Architecture

Part 2:

Building simple processors

Simon Hollis (simon@cs.bris.ac.uk)

How to use lectures

- The material for the unit will be that covered in lectures.
- My style will be to talk and write things on the board/OHP.
- You may wish to note things down, so bring a pen!
- Rather than producing printed notes, you can find all the discussed topics in detail in the recommended course textbooks and later on, the course wiki.

An introduction

- This part of the course talks about the creation of a functioning computing system.
- It is based on the logical components and design ideas you have seen in the first part.
- We will “black box” the low-level details, and deal instead with functional blocks.

Context for this part

- This part aims to deliver the essentials of computer ***architecture*** from the basics up.
- We will treat the material from a fairly abstract point of view at first, but one that is realised by all modern machines.
- By the end of the course, you will be able to understand real contemporary machines
- The full complexity of the hardware architecture comes in the assignments and later courses (e.g. Advanced Computer Architecture, Embedded Systems Integration).

Material overview

Dan

Introduction to logic
Introduction to numbers and arithmetic
Sequential and combinatorial functions.

Simon

A hands-on introduction to processor design
An introduction to instructions and assembler.

David

Hardware and software interaction.
Compilers
Memory and I/O

Assessment

This course has lots of formative assessment:

(Weekly hardware labs with worksheets)

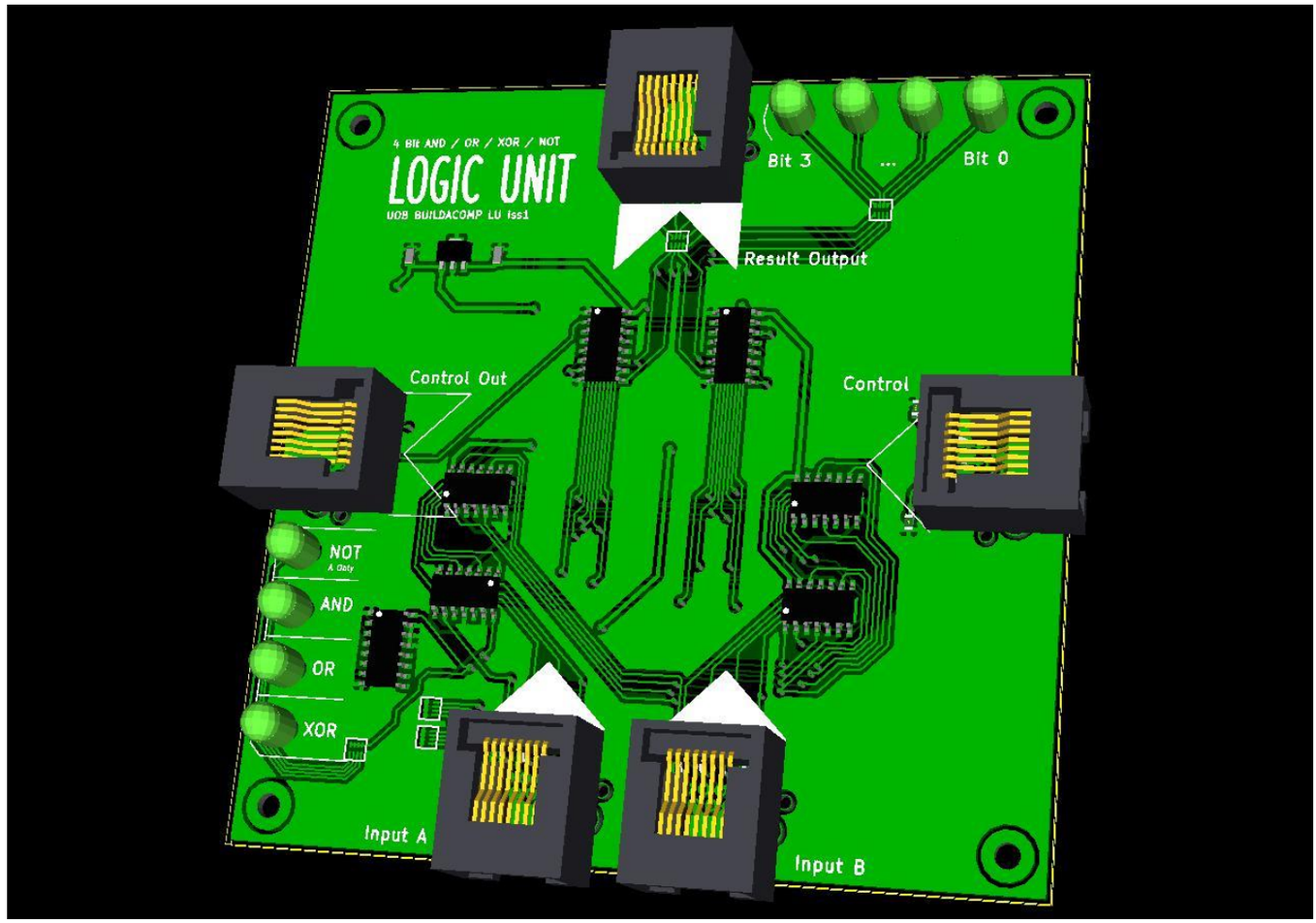
And, two measuring assessment:

(2x viva)

Our target

By the end of this part of the course, you will know how to combine blocks of real hardware to make a computer processor, whilst understanding how it works.

We'll be using UoB developed teaching modules in combination with labs to make this happen.



Current course information

- The most current, and canonical information is always that posted on the course website.

COMS12200 Topic 1

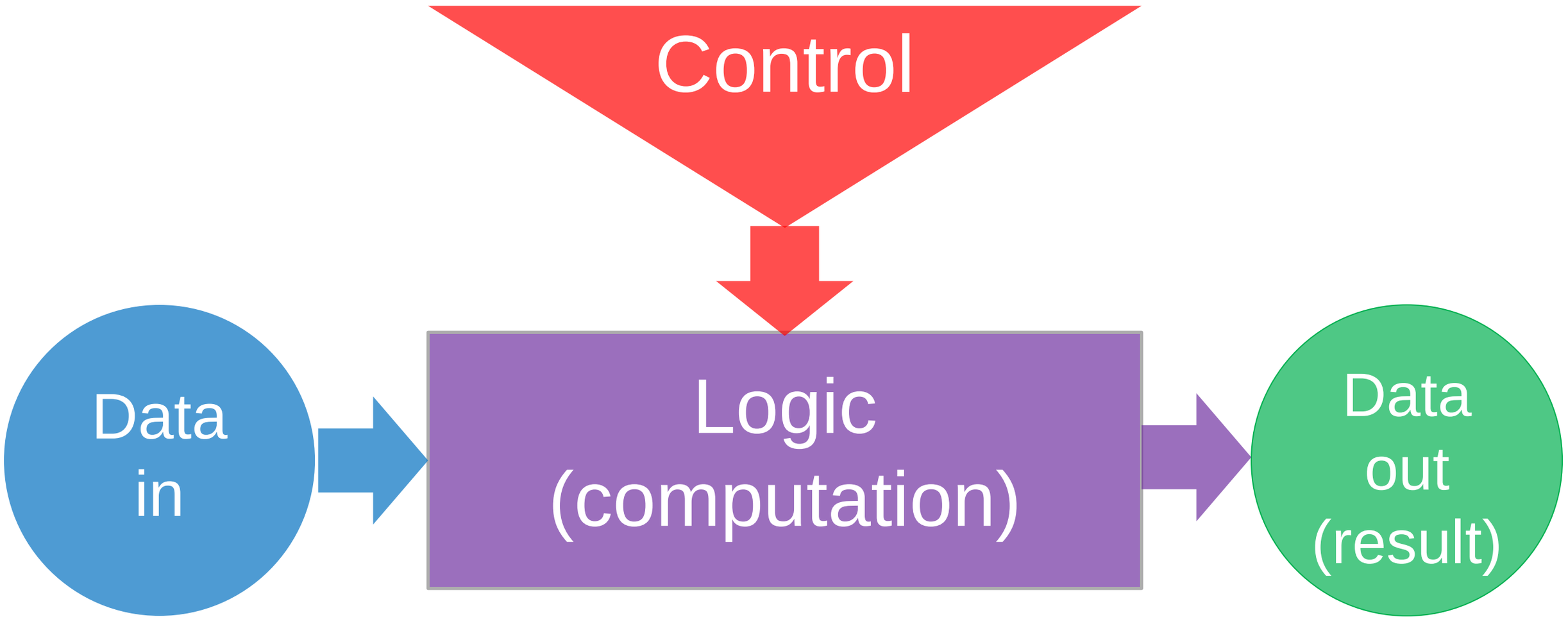
DATA, CONTROL AND INSTRUCTIONS

Data and control

- You can think of a processor's function as being dictated by two separate influences:
  - The ***control*** information (which tells it what to do)
  - The ***data*** information being operated on (which tells it with what to produce the result)
- These two influences form two paths into the processor logic

Data and control

In Dan’s part of the course you saw the notion of data and control paths



What is data?

- Data is simply some information. It can be *any* value.
- How it is stored is unimportant.
- How it is formatted is unimportant.
  - “0110” is data;
  - “Hello” is data;
  - “☺” is data
- Data stores information and forms inputs to, and outputs from, calculations.

Where does data live?

Data lives in storage elements.

There are many different elements in a modern processing system.

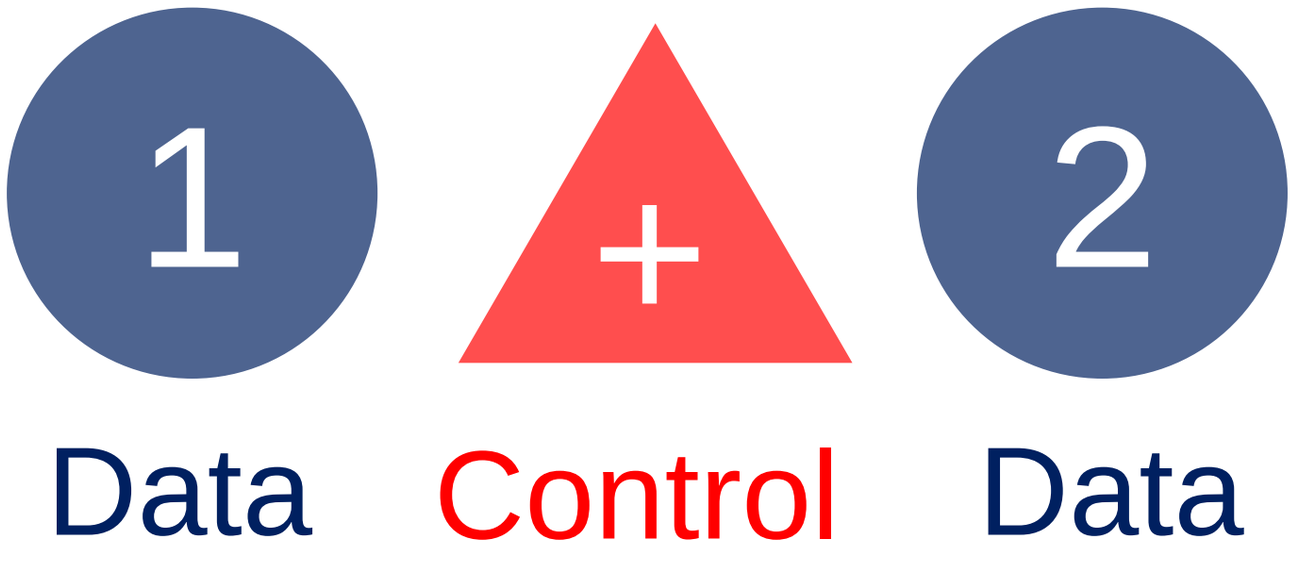
In this course, we'll concentrate on two: memory and registers.

Both can be treated as 'black boxes'.

Processor instructions always operate on registers and sometimes memory too.

Control information

- Control is also information, but its use is different.
- It specifies what must be done
- It is applied to a system (not consumed by it) and its values do not appear in the output data



Control and instructions

- You've seen how to input control information before, in the calculator example.
- We've encoded choices, such as 'ADD' with a selection of specific control signals being activated and deactivated.

Control and instructions

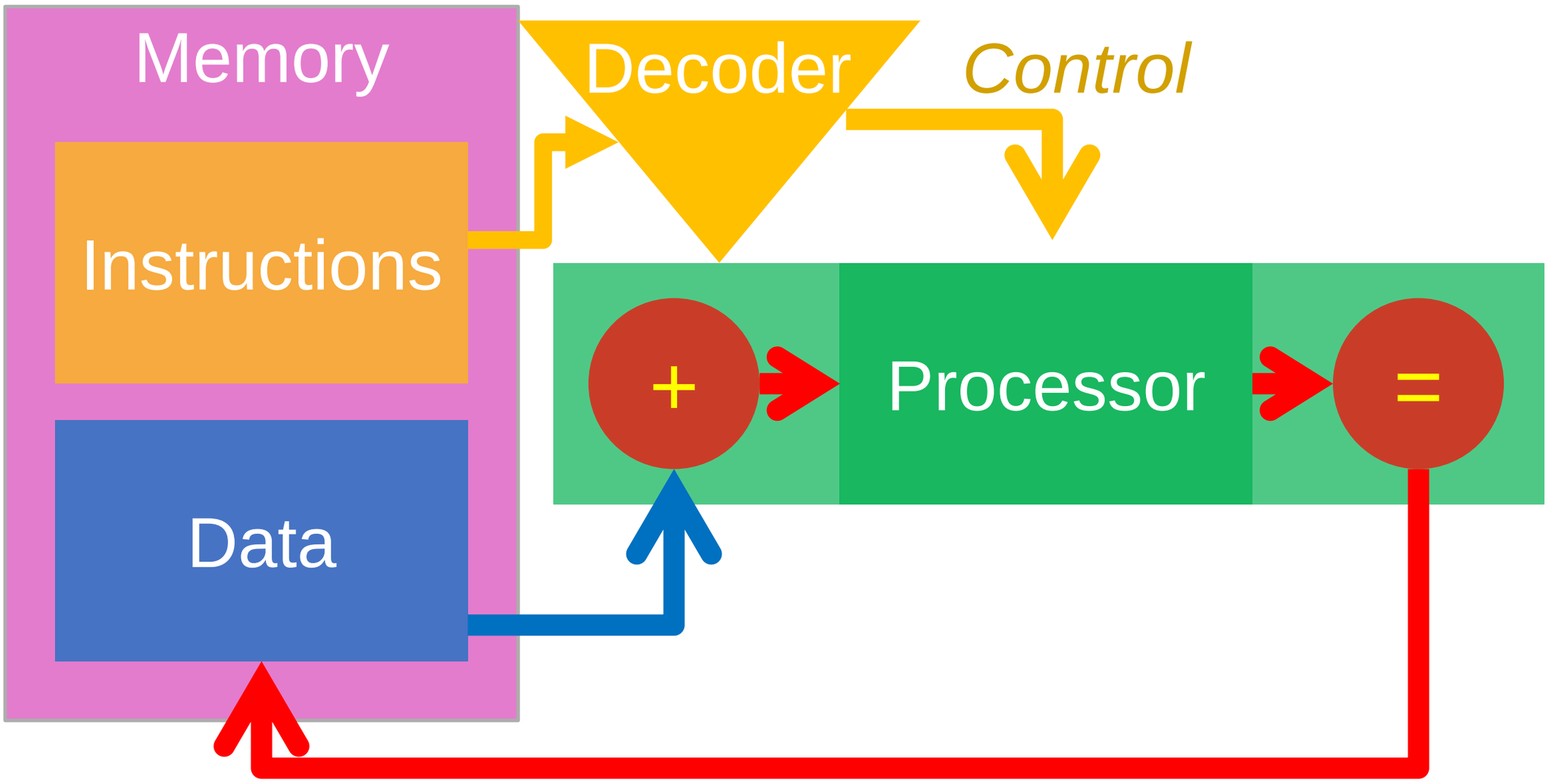
- In small-scale systems, this is OK, but can you imagine how many switches need to be flipped to choose between all of the functionalities of a modern processor?
- We need a way of condensing the control information.
- We need a way of programming the machine with it.
- The solution is to use ***instructions***

Instructions at a high level

- E.g. If an instruction = “A”, the processor should do “X”; if an instruction = “B”, the processor should do “Y”. And, so on.
- At the high level, we can treat them as abstract constants, whose value causes different processor behaviour.
- Instructions need to be *decoded* before they can be used.

How it works

- Before moving on to how a computer operates, we need to understand its inputs and outputs.



What is in an instruction?

- An instruction is also information.
- However, it has a defined purpose: to specify an exact amount of work to be done by a processor.
- This leads it to have a specific form and formatting (more in “ISAs” section of this course).
- Only a sub-set of all possible values are valid.

Instruction encoding

- Instructions allow us to encode the control information that we need to control a computing system.
- The key is to use a unique code per unique function.
- The specific encoding is called an ***op-code***

Some op-codes

Example: here's how several different processor ISAs use different op-codes to encode the same instruction (to add together two numbers)

ISA (Processor instruction set)	Encoding of 'ADD' (binary)
ARM	00110
MIPS	100000
Intel x86	00000000
PIC	000111

Decoding

- There are many different encodings of instructions for the same meaning.
- Systems need an architecture-specific decode module to determine the meaning of the op-code as an instruction and activate the relevant control lines.

Example decoder

Input op-code	Instruction
00	‘ZERO’
01	‘ADD’
10	‘SUBTRACT’
11	‘MULTIPLY’

There are many possible logical structures for a decoder.
e.g.

1. Combinatorial
2. MUX-based
3. Lookup-based

In class examples of structures

Summary

Today we have seen:

- Data and control relating to a computational structure.
- What control information looks like.
- An introduction to instructions.
- How instructions are encoded as op-codes.
- How op-codes can be decoded.