A Computer program is a collection of bit sequences that represent actions that a computer should undertake.

The number and type of such actions is dictated by number and type of components that make up the computer and the ways in which these components can interact.

Since we, as programmers, dictate which actions should take place, and when they should take place, we see these actions from our perspective and call them instructions.

Choosing an appropriate set of components and a limited number of ways that these components can interaction — while being able to do the required work is part of the art, science and engineering of computer architecture.

In our studies so far, we have designed and virtually constructed all of the component needed for a functioning computer.

We will now connect them together and will design and use a simple set of instructions to exercise them.

Let's first recap to list the Component that we have created:

An 8-bit ripple carry Adder

An 8-bit ripple carry Subtractor

Multiplexers

De multiplexers

Controllers

8-bit Registers

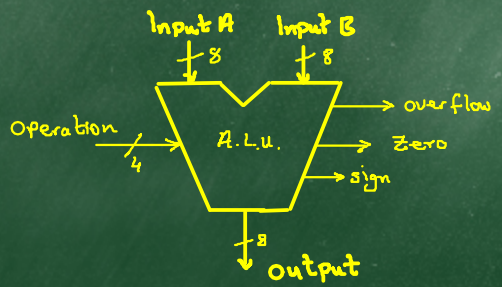
12 byte SRAM

ALU

In addition to arithmetic operations we can also perform logical operations on bit sequences. For example, we can get the bitwise OR of two sequences, etc.

The arithmetic and logic operators are abstracted away into a black-box known as the Arithmetic Logic Unit (the ALU)

The ALU is often represented as follows:



The inputs and output of our ALU are all 8-bits. The inputs will typically come from values stored in registers and the output will typically also be directed to a register.

The operation input specifies what the ALU should do with the inputs: add, subtract, Bitwise AND, Bitwise OR, etc.

The single-bit outputs: overflow, zero and sign (there may be others also) are stored in a special register called the Status Register. These flags reflect what happened when performing the operation.

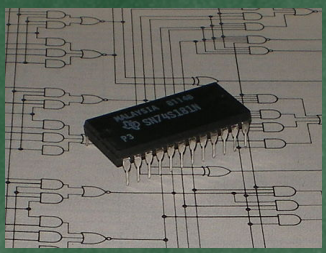
So, if the was a carry out when performing addition, the overflow flag is set;

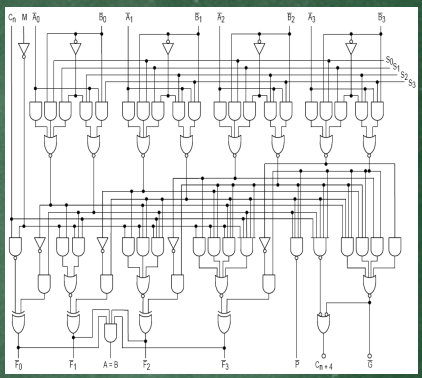
If the result of the operation was zero, the zero flag is set;

If the operation resulted in a sign change, the sign flag is set.

Aside

The first ALU to be created as an Integrated Circuit was the Intel 74181 in 1971.





It was a 4-bit ALU and Contained 70-Gates.

Control Unit

We have seen how to Construct Controllers that active various outputs based on their input values.

Our Computer also uses such a device, it takes in a number of Inputs (the op Code bits of an instruction) and it activates and deactivates other Components in an orchestrated manner to Cause values to move between these Components. It also 'tells' Certain Components how to behave. This device is Call the Control Unit.

For example,

it will 'tell' the ALU what operation to perform;
it will 'tell' registers and memory when to read or write;
etc.

We will see that each program instruction (think of Samphire) will be processed by the Control unit in a number of steps.

Each step is called a **micro instruction**.

The start and End of each microinstruction is determined by a special device that oscillates between 0 and 1 repeatedly at a fixed Speed. This device is called the **clock**.

In general, the faster the clock, the faster our computer will execute instructions. But there is a limit, determined by the amount of time it take each Component to act.

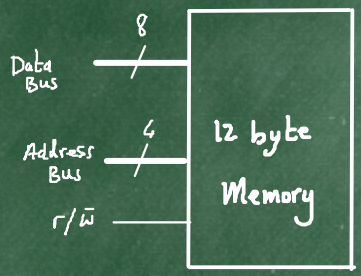
If we go faster than that limit, the system will fail to act in a predictable manner.

In Some situations, faster is not always better – Speed Consumes Power.

Memory

We saw in Lecture 20 how we might construct SRAM.

We'll abstract it again here, showing only its main features:



Memory is used to hold both data and instructions.

It is typically partitioned into areas for each.

User programs may read and write data to/from memory but typically program instructions are only read from memory.

When they are being read, instructions move over the 'data bus'.

Registers

Registers are used in our Computer for a variety of purposes. There are two main categories of use:

(i) System Registers

(ii) Data Registers

System Registers

The registers are used to support the various actions needed to execute each (micro) instruction

They include the following:

The Program Counter (aka instruction pointer).

This register is used to hold the address of the next instruction to be executed.

The Instruction Register.

This register holds an instruction after it has been fetched from memory.

The bits of this register holding the Opcode of the instruction are connected to the control unit.

The Status Register

This register is used to hold the values of the status bits (flags) generated by the ALU.

These flags can be used to determine what should happen next.

Recall from Samphire

```
jz loop
```

– Jump if the zero flag is set to the instruction labelled 'loop'.

There may be other system registers, depending on the capabilities of the specific microprocessor – stack pointer, etc. But for our purposes this is enough for the moment.

Data Registers

Data Registers are used to

(i) hold operand and result data

(ii) act as temporary waiting areas as data (instructions) are transported from one component to another. For this purpose, there are often called buffers

In our simple machine, we will use just two operand registers which we will call A and B.

We are now in a position to connect all our hardware components together and in the process create a Central Processing Unit, a CPU: