A Brief Clarification

- Some of you seem to be confused as to what data is – whether electrons have memory...
- Data with respect to our course is the *information* stored within the memory of a system (Memory will be explained later. For the time being it is just a storage space.)
- *One bit of data means a logical 0 or 1* (for our understanding)
- If we say that the output of a logic gate is *1*, it means it is providing (not necessarily storing) a bit of data whose state is *1*.
- If you measure the voltage at its output it will be around 5V (or 3V). Likewise, if it is *0*, then the voltage will be 0V indicating that the data is *0*.
- So if a conducting line is somehow held at 5V (or 3V) we can say that it is carrying a bit *1* or its state is logic *1*. (Else if it is 0V, then bit *0* logic *0*)
- So deep down it means that if a bit *0* is being stored in the memory, then that particular point in the memory where we are storing the bit, is somehow being held at 0V. If the data stored is *1*, then the voltage at that point is somehow held at 5V (or 3V).
- Conceptually if I wish to store *01001000* (1 byte) then I would need the memory to hold the following voltages at 8 different points within as:

0V	5V	0V	0V	5V	0V	0V	0V	0V

- The above 8 blocks storing 8 bits is often referred to as an 8-bit *Register*

Coming soon: More on Memories!

Interpreters

Compilers

Ram goes to school.
Rashid goes to school.
Nancy goes school.
Ram happens to be an intelligent guy.
Rashid is very good at studies
Nancy is a brilliant student.

Compare this with:
Reading a story sentence by sentence and translating it to another language (INTERPRETING)
versus
Reading the whole story, understanding it and then writing the story in the other language without looking at the original. (COMPILING)

Friday, March 25, 2022

Interpreters	Compilers
Translates the source program statement by statement into machine code.	Scans the entire program and then translates the whole of it into machine code at once.
Since only one statement is being dealt with at a time, an interpreter takes very less time to analyze the source code. But, the overall execution time of the entire program is effectively large - i.e. execution is slow.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. So it does not need much of memory – in short it is efficient when it comes to use of memory.	A compiler always generates an intermediary object code. Later it may need to link many aspects of the intermediary code. Hence more memory is needed.
Keeps translating the program continuously line by line till the first error is confronted. If any error is detected, it stops working and hence debugging is easier.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder.
E.g. Python (Find more examples)	E.g. C (Find more examples)

Levels of Abstraction

High Level Language Program (e.g., C)

```
x = 3;
y = 4;
z = x + y;
```

Compiler

Assembly Language Program

```
MOV A, 03H
MOV B, 04H
ADD B
MOV C, A
```

Assembler

Machine Language Program

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Machine Interpretation

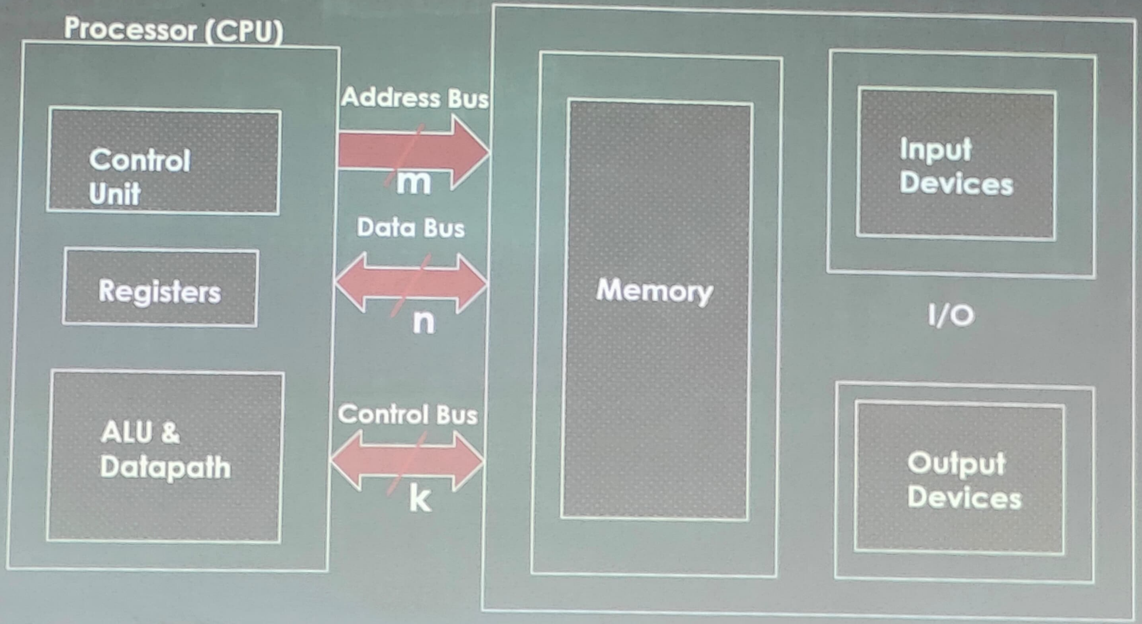
Data path Transfer Specification

IR ← Imem[PC]; PC ← PC + 1

NB: IR is the Instruction Register and PC is the Program Counter, Both these are within the CPU

Computer H/W Organization

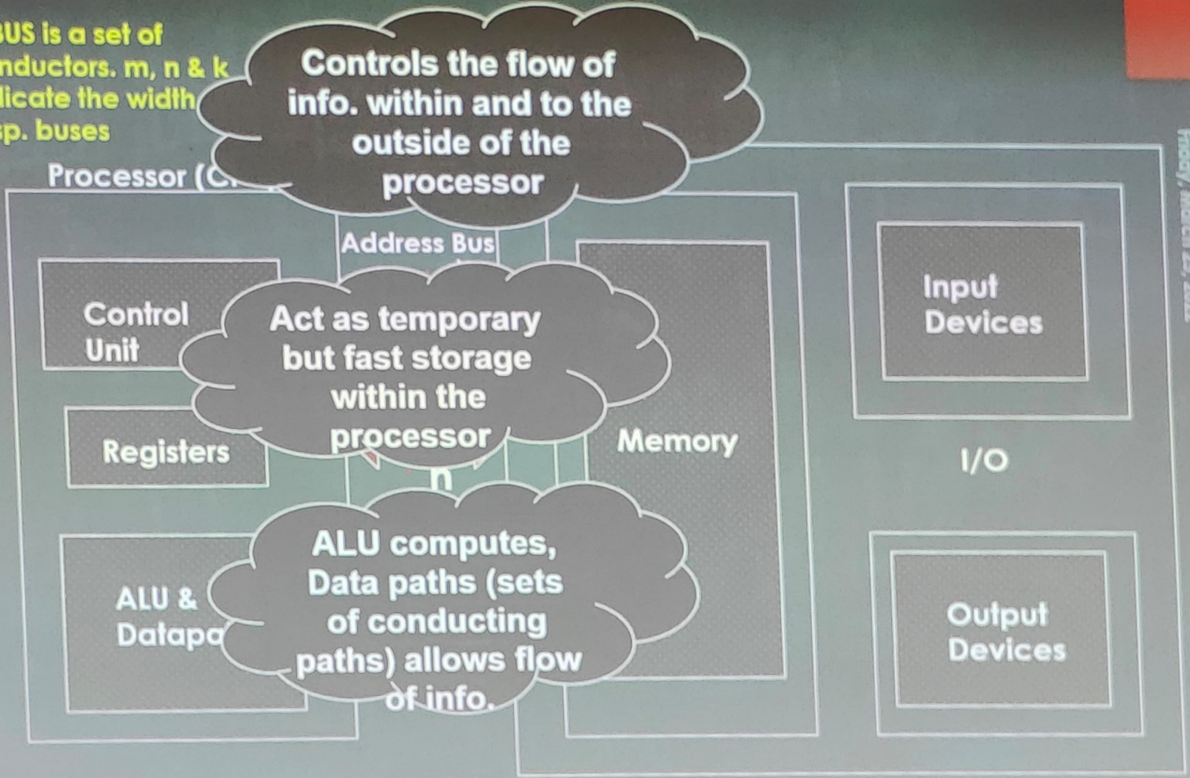
A BUS is a set of conductors. m, n & k indicate the width of the resp. buses



Friday, March 25, 2022

Computer H/W Organization

A BUS is a set of conductors. m, n & k indicate the width resp. buses



Friday, March 25, 2022

Memory Memorabilia

- Stores data and facilitates its retrieval at a later stage
- Conceptually, a computer memory is simply a collection of locations where information can be stored as bits
- Most often, memory is byte-addressable
- This means the memory is divided into *bytes* (8-bit quantities) each identified by a unique address
- Generally, bytes are addressed sequentially, beginning with the address 0.

Address	Byte 7 6 5 4 3 2 1 0
0	| ... | | |
1	80
2	45
3	67
...	
1004	22
1005	9
1006	25
...	

Size of Memory

Generally specified as:

No. of words x No. of bits/word

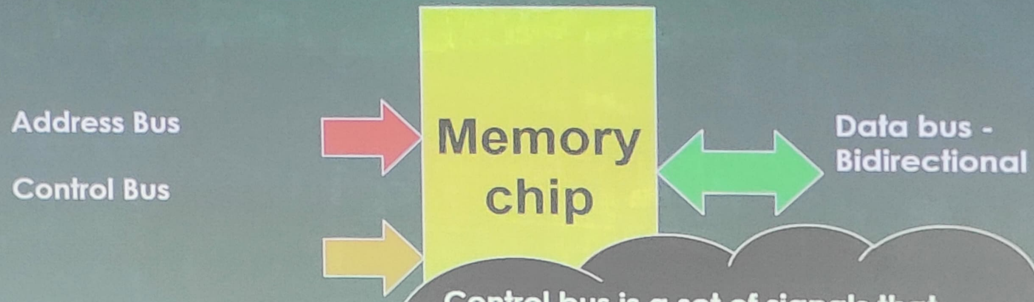
E.g. 1024x1 = 1Kbit → There are 1024 words each of 1 bit length

1Kb

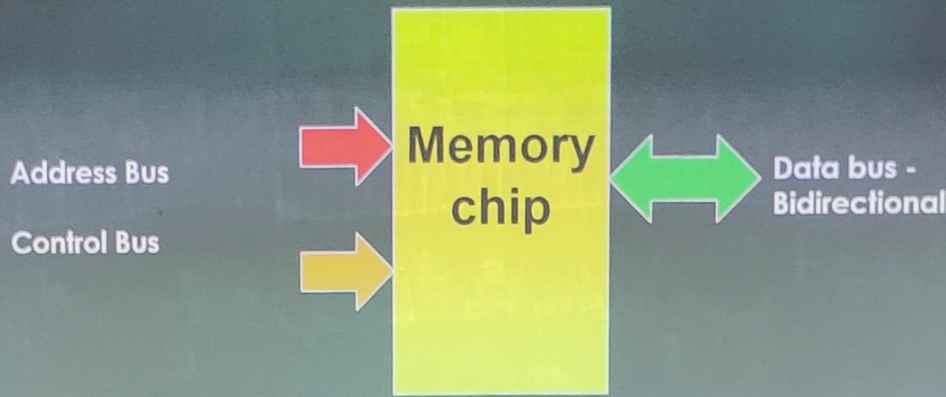
2048x8 = 2KByte → There are 2048 words each 8 bits in length

2KB

Memory Subsystem



Control bus is a set of signals that help maintain a protocol between the CPU and memory and other devices. The signals indicate a variety of operations the CPU wishes to do on these external entities. (Like Read, Write, Ready, Wait,....)

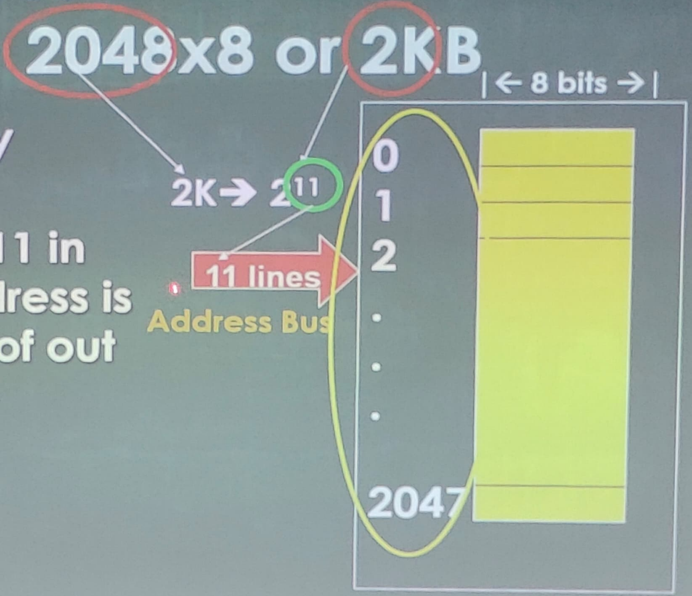


- 2KB → 2048 x 8 (Has 2048 words each 8-bit wide)
- For every word to have a unique address this memory needs to have 2048 unique addresses (nos.)
- The number of bits required to generate 2048 unique addresses is thus 11.
- Why is that so? B'cos $2^{11} = 2048$
- So if we have *n* bits we can generate 2^n unique addresses.
- It also means that if a memory has say 2048 words, then it has an address bus whose width (no. of lines within this bus) is 11

E.g. Address Bus

There are 2048 memory locations.

Address lines are thus 11 in number as a 11 bit address is required to select one of out of 2048 locations in the memory



Friday, March 25, 2022

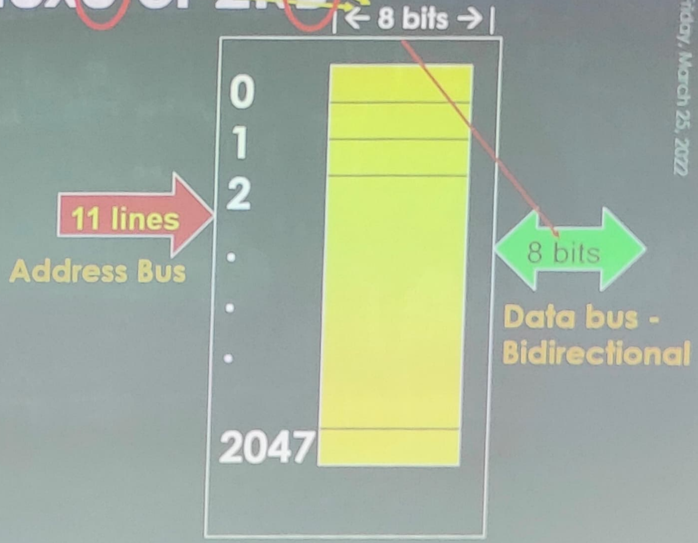
E.g. Address Bus

2048x8 or 2KB

Each location within this memory can store 8 bits.

So we need to read or write 8 bits of information at a time.

The Data bus therefore has 8 lines.



Friday, March 25, 2022

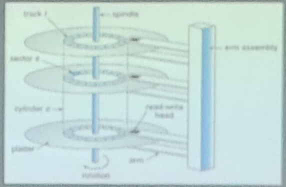
Taxonomy Memory: Based on Access Method

Random Access Memory (RAM)

- Individual addresses identify locations exactly
- Access time is independent of location or previous access
- e.g. RAM (we will come to this later)

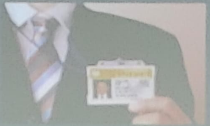
Sequential Access Memory

- Start at the beginning and read through in order
- Access time depends on location of data and previous location
- e.g. magnetic tape



Direct

- Individual blocks have unique address
- Access is by jumping to vicinity plus sequential search
- Access time depends on location and previous location
- e.g. disk



Associative

- Data is located by a comparison with contents of a portion of the store
- Access time is independent of location or previous access