# The Central Processing Unit.

18 January 1999

By the end of this session, the class will understand enough of the internal workings of the CPU to comprehend the meaning and use of terms such as: binary numbers, byte, kilobyte, megabyte and WORD.

## *The motherboard.*

In Microcomputers, the motherboard holds the circuitry, transistors, chips that work together with the CPU to get things done.

## *The CPU - The internal workings*

Computers do not think like you or me, but they “compute”.

The CPU is an electrical device so the only states that it understands is ON and OFF. This is represented when we discuss computer thinking as 1 or 0.

This does not give much room for a complicated view of life, using 1’s and 0’s, this is called the binary system.

The Central Processing Unit (CPU) directs all the activities of the computer. It can only follow instructions that it normally receives either from ROM or from RAM. In following these instructions, the CPU guides the processing of information throughout the computer.

## *Binary Numbers*

Early computer systems used electrical switches and when electrical switches were replaced by less mechanical devices such as vacuum tubes, than the transistor, the integrated circuit, the concepts of switching on and off remained with computers but a representation of the on/off behaviour of computers had to be made.

The binary number system, where a zero symbolises no electrical current (OFF) and a one represents electrical current exists (ON) developed and became the standard means of representing internal computer workings. By combining a series of these 0’s and 1’s (OFF/ON), the computer is capable of representing a number of complex things.

To understand the binary number system, let us first review the decimal counting system.

The Decimal number system can be represented as in Table 4.1 The place of the numerical digit has significance (value). The decimal system has units for a digit in 0 through to 9, which adds up to being 10 numbers. Therefore when you write a decimal number down it would be represented like this:

| Number | |
|---|---|
| 13 | is equal to 1 set of ten and 3 units (1 * 10 + 3 * 1 = 13) |
| 27 | is equal to 2 set of tens and 7 units (2 * 10 + 7 * 1 = 27) |
| 302 | is equal to 3 set of hundreds, 0 set of tens and 2 units (3*100 + 0*10 + 2*1 = 302) |

or they could be represented such as in Table 4.1.

Table 4.1 A representation of the place value of decimal numbers

| | 1,000,000 | 100,000's | 10,000's | 1,000's | 100's | 10's | Units | | |
|---|---|---|---|---|---|---|---|---|---|
| | x $10^6$ | x $10^5$ | x $10^4$ | x $10^3$ | x $10^2$ | x $10^1$ | x $10^0$ | | |
| | | | | | | 1 | 3 | = | 13 |
| | | | | | | 2 | 7 | = | 27 |
| | | | | | 3 | 0 | 2 | = | 302 |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

The Binary number system uses only the 0 and the 1, therefore the UNITS can only be either 0 or 1.

Therefore when you write a decimal number down it would be represented like this:

Number

13 is equal to 1 eights, 1 fours, no twos and 1 unit
(1*8 + 1*4 + 0*2 + 1 * 1 = 1101)
27 is equal to 1 sixteens, 1 eights, no fours, 1 twos, 1 units
(1*16 + 1*8 + 0*4 + 1*2 + 1*1 = 11011)
302 is equal to 1 two-fifty-sixes, no one-twenty-eights, no sixty-fours, 1 thirty-twos, no sixteens, 1 eights, 1 fours, 1 twos and no units
(1*256 + 0*128 + 0*64 + 1*32 + 0*16 + 1*8 + 1*4 + 1*2 + 0*1 = 100101110)

or they could be represented in a table:

## *In Class Exercise:-*

Translate the following Bytes to Decimal

| Byte | Decimal |
|---|---|
| 00000001 | 1 |
| 00001001 | ? (9) |
| 00001000 | ? (8) |
| 00000100 | ? (4) |

Translate the following Decimal values to Bytes

| Decimal | Byte | |
|---|---|---|
| 6 | ? | 00000110 |
| 17 | ? | 00010001 |
| 50 | ? | 00111010 |
| 200 | ? | 11001000 |

Homework Review Questions:-
Translate the following Decimal to Binary

| Decimal | Binary |
|---|---|
| 13 | 1101 |
| 21 | 1 0101 |
| 207 | 1100 1111 |
| 1994 | 111 1100 1010 |
| 45 | 10 1101 |
| 73 | 100 1001 |

Translate the following Binary to Decimal

| Binary | Decimal |
|---|---|
| 100 1110 | 78 |
| 1101 0000 | 208 |
| 10 1010 | 42 |
| 1 0001 0100 | 276 |

## Bits and Bytes

The Unit of binary "0" or "1" is called a bit (binary digit). Since a single bit by itself gives little information, bits are usually grouped together as shown in our numerical calculations above.

If a CPU made its calculations using a group of 4 bits together

| 8 | 4 | 2 | 1 |
|---|---|---|---|
| __ | __ | __ | __ |

We get a maximum number of 15 .

If we increase the numbers of bits the CPU uses to a grouping of 8 bits:

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| __ | __ | __ | __ | __ | __ | __ | __ |

We get a maximum number of 255.

If we increase the number of bits the CPU uses to 16 groups of bits:

| 32,768 | 16384 | 8192 | 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ | __ |

We can get a maximum number of 65,535.

## Why is this important? (1) Calculations

From the above examples, we can see that the bigger the group of bits a CPU uses, the bigger the numbers that it can directly calculate (without any fancy tricks). Not only are bigger numbers available, but we can already see that mathematics will be more accurate the more bits the CPU uses.

Many machines are referenced as 8 bit, 16 bit, or 32 bit. This refers to our discussions above on how the computer groups its bits together.

## Why is this important? (2) Memory

We know that the size of the bit grouping defines how big a number the CPU can make-up.

Most of the memory (primary storage) that the CPU uses is external to the CPU. The computer memory is arranged like a table of ON/OFF switches.

These memory "blocks" are organised to have the same group size of bits as that being used by the CPU, and the CPU references, gets to the memory, addresses the memory just like a row of bits that it will request.

Therefore the 4 bit CPU can address/access 15 rows of 4 bits from the memory table.

The 8 bit CPU can address/access 255 rows of 8 bits from the memory table.

The 16 bit CPU can address/access 65,535 rows of 16 bits from the memory table.

The amount of memory that the CPU can look up (address) is called the "Linear address space" of the CPU.

## Homework Review Questions

What is the maximum number of rows of datum can the following bit grouping "address"?

| | | |
|---|---|---|
| 32 bit | 4,294,967,295 | 4 Gigabytes (billions of bytes) |
| 64 bit | 18,446,744,073,710,000,000 18 | |

## Why is it important? (3) The WORD

The CPU bit grouping size is also referred to as the CPU's WORD size. Another way to think of the bit grouping size (WORD) is as the number of bits that constitute (make-up) a common unit of data as defined by the CPU.

Common WORD lengths are 8 bits (for some microcomputers and Sega, Nintendo machines), 16 bits, 32 bits, and 64 bits.

The WORD size, apart from the already discussed:
Better numbers, calculations
Reference/address larger memory tables
Will also allow the CPU to transmit (send and receive) more data at a time.

A 16 bit CPU can transmit 16 bits of information in one electricity burst, where an 8 bit CPU can only send half (8 bits).

It all sounds simple enough to use the largest word possible, but the difficulty is that the physical hardware has to support the word size. For example, if the CPU word size is 32 then the lines between the CPU and the memory has to support sending through 32 separate bits of data with each instruction. That is, with each "burst" of electricity, the CPU can send through the motherboard 32 bits simultaneously.

## *Definitions:*

When we group bits together they have been given names by the computer industry.

One binary digit is a BIT
Eight bits is a BYTE (pronounced "bite")
Four bits is half a byte, so we call it a nibble.

Most references to computers use the number of "bytes" as a measure for the computer's memory (primary storage) capacity and storage (secondary) capacity.

A kilo in decimal numbers is equal to 1,000 or $10^3$, the closest number using the 2x is 1,024, and this is $2^{10}$. In computing terminology, 1,024 bytes is 1 Kilo Byte or 1 K.

Therefore a 256K machine would have 256 * 1,024 bytes of storage, or 262,144 bytes of storage.

## In Class Exercise

How many bytes are in the following numbers?

| | |
|---|---|
| 512K | 524,288 |
| 640K | 655,360 |
| 720K | 737,280 |

Using the same principle, a mega (or million) in decimal numbers is equal to 1,000,000 or $10^6$. The nearest approximation for the power of 2 is 1,048,576 or $2^{20}$. Therefore, 1 mega byte is equal to 1,048,576 bytes or 1 M.

## In Class Exercise

How many bytes are in the following numbers?

| | |
|---|---|
| 4 M | 4,194,304 |
| 6 M | 6,291,456 |
| 10 M | 10,485,760 |

## ASCII - American Standard Code for Information Interchange

Because the purpose of using computers is not to just throw around numbers, the byte was also used to represent the alphabet, including special characters. The Americans were the first to really work on a standard, which has been universally accepted in the English speaking world, called ASCII for American Standard Code for Information Interchange.

Instead of using a full byte to represent data, ASCII uses only 7 bits. By referring to the above discussion, you will realise that the combination of 0's and 1's has a maximum value of 64 + 63 = 127.

Table 4.2 shows how ASCII represents the alphabet capital letters:

Table 4.2 ASCII representation of capital letters

| | BYTE | dec. | | BYTE | dec. |
|---|---|---|---|---|---|
| A | 100 0001 | 65 | N | 100 1110 | 78 |
| B | 100 0010 | 66 | O | 100 1111 | 79 |
| C | 100 0011 | 67 | P | 101 0000 | 80 |
| D | 100 0100 | 68 | Q | 101 0001 | 81 |
| E | 100 0101 | 69 | R | 101 0010 | 82 |
| F | 100 0110 | 70 | S | 101 0011 | 83 |
| G | 100 0111 | 71 | T | 101 0100 | 84 |
| H | 100 1000 | 72 | U | 101 0101 | 85 |
| I | 100 1001 | 73 | V | 101 0110 | 86 |
| J | 100 1010 | 74 | W | 101 0111 | 87 |
| K | 100 1011 | 75 | X | 101 1000 | 88 |
| L | 100 1100 | 76 | Y | 101 1001 | 89 |
| M | 100 1101 | 77 | Z | 101 1010 | 90 |

*Table 4.3 Convert binary to ASCII text*

| | | | |
|---|---|---|---|
| 100 1000 | H | 100 0110 | F |
| 100 0101 | E | 101 0010 | R |
| 100 1100 | L | 100 1001 | I |
| 100 1100 | L | 100 0100 | D |
| 100 1111 | O | 100 0001 | A |
| 100 0011 | C | 101 0011 | S |
| 100 1111 | O | 101 0100 | T |
| 100 1101 | M | 101 0101 | U |
| 101 0000 | P | 100 0100 | D |
| 101 0101 | U | 100 1001 | I |
| 101 0100 | T | 100 0101 | E |
| 100 0101 | E | 101 0011 | S |

*Table 4.4 Convert text to ASCII binary*

| | | | |
|---|---|---|---|
| T | 101 0100 | your name. | |
| O | 100 1111 | | |
| D | 100 0100 | | |
| A | 100 0001 | | |
| Y | 101 1001 | | |
| Q | | S | |
| U | | P | |
| E | | O | |
| E | | R | |
| N | | T | |
| S | | S | |

## In Class Exercise:-

Determine the binary ASCII code for the words in Table 4.3

Determine the ASCII letters from the binary in Table 4.4

## Homework Review Questions

Write the ASCII text for the following:
Sports Day
Queen
Salote
Mailefihi
Broom

# Control, A/LU and Memory

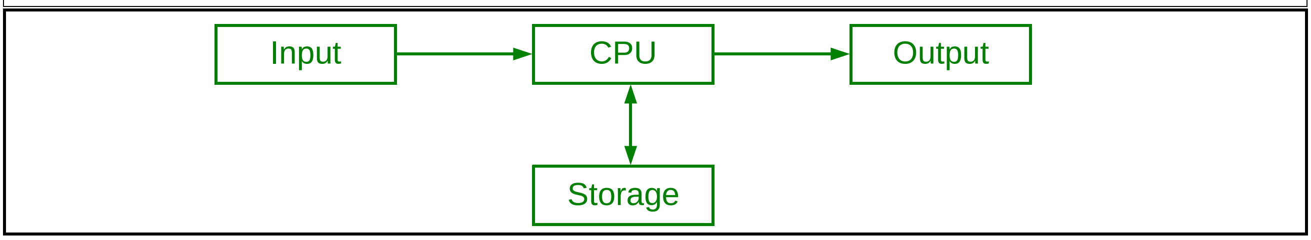
So now, we know part of what is used in the CPU number process, but that doesn't really help us understand how people got programs working.

The Block Diagram is a little simplistic, as Input Devices often receive output from the Processor. Likewise, Output devices (like screens and printers) sometimes send information (input) to the Processor.

The difficulty we've had just translating a few things back and forth into BINARY is an example of how hard it was in the early days of computers for people to program computers. Even for the simplest of things it was difficult, and there were many mistakes.

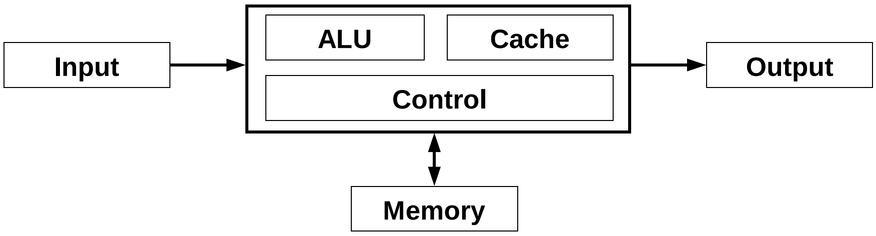
Diagram 4.1 shows the data flow inside a computer, notice that all information first flows through the CPU. Because one of the jobs of the CPU is to control the order in which tasks are completed it is often referred to as the "brain" of the computer. The CPU only executes tasks according to the instructions it has been given, it cannot think for itself.

*Diagram 4.1 - Classical Block Diagram of Data flow in a computer*



Most computers have two types of memory chips, ROM and RAM. Read Only Memory (or ROM) often contains the most basic operating instructions for the computer. It is made a permanent part of the system and cannot be changed. The instructions in ROM enable the computer to complete simple jobs such as placing a character on the screen or checking the keyboard to see if a key has been pressed.

A Discussion Model: The Central Processing Unit




Random Access Memory or RAM is temporary memory where data and instructions are stored. Data stored here can be changed or erased. When the computer is first turned on this part of memory is empty and when turned off, any data it stores is lost. Because RAM storage is temporary, computers use disks as auxiliary memory storage. Before turning the computer off, the data is stored in RAM can be saved on a disk so that it can be used again at a later time.

The diagram shows that the CPU is made up of at least three components, the Control, the A/LU (Arithmetic Logic Unit) and some memory cache. This is a sample of the minimal things within a Central Processing Unit, in some CPUs there are more capabilities.

## The Control

The “Control” does what it is called, and says what is going to happen. The Arithmetic part of the A/LU knows about mathematical things like add and multiply. The Logic part of the A/LU knows about logical things like AND, OR, NOT.

All electrical activity is adjudicated, apportioned by the CONTROL. Like the traffic police, the CONTROL decides which action will occur. For example, on the road the police decides which lanes will travel and which lanes will stop. For computers, the CONTROL decides which electrical signal coming into the CPU will get through first, and which action by the CPU will continue next.

The Arithmetic Logic Unit, or ALU, is the part of the CPU where the “intelligence” of the computer is located. It can perform only two operations. It can add numbers and compare numbers.

## The Arithmetic Unit

The arithmetic part of the CPU does the maths. Most CPUs have at least the ADD function and the MULTIPLY function. It is not necessary to have the MINUS function because you can subtract a number by adding the negative of that number.

e.g. $x - y = x + (-y)$

Likewise, it is not necessary to have the divide function as division can be achieved by multiplying the inverse of a number.

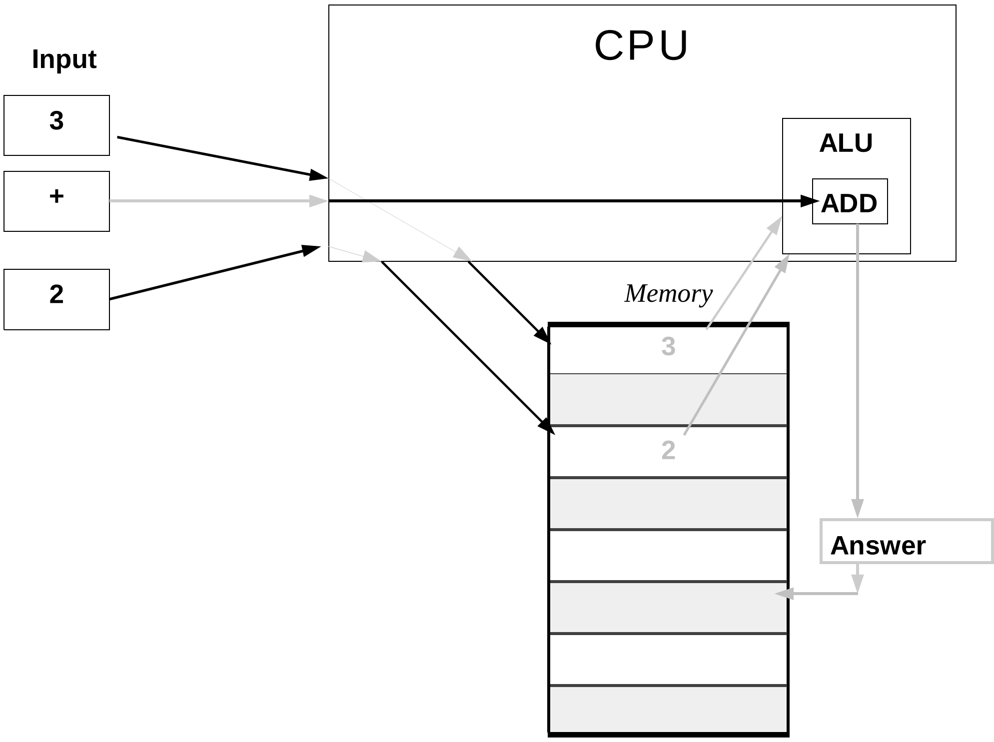
**e.g. x / y = x * 1/y**

To demonstrate a CPU's action in adding two numbers, let us take a look at the steps the CPU goes through.

Steps:

- The numbers to be added up are put into two separate memory locations,
- You tell the control to tell the Arithmetic Unit to add the two numbers in the two memory locations together.
- You tell the ALU to stick the answer into memory location X.

Now, if you actually want to make use of this number, you have to look at the memory location X for what the answer is.




The advantage of the computer is that it can do all of this very quickly, and programs have already been written to make all of this easier for us.

## The Logic Unit.

Logic is being able to associate a result from previous experience. The importance of the Logic Unit is the ability it gives the CPU to make a decision (based on the instructions provided to it) depending on the results of a comparison of results from data it has received. The ability to create machines able to make decisions is a defining point in the development of computers, transitioning the machine to being more than just a fancy calculator to a tool that can perform numerous other tasks.

Example:

If you must have a valid visa and a transport ticket to travel to New Zealand then :

- If you have a valid visa but NO transport ticket then you cannot go to New Zealand.
- If you do not have a valid visa but have a transport ticket to New Zealand you cannot go.
- If you have a valid visa and a transport ticket to New Zealand then you can go to New Zealand.

*Table 4.3 Logical Comparisons*

| Comparing the value | Using the Logical Comparison Operator | With the following | Results in the following |
|---|---|---|---|
| TRUE | AND | TRUE | TRUE |
| TRUE | AND | FALSE | FALSE |
| FALSE | AND | FALSE | FALSE |
| TRUE | OR | TRUE | TRUE |
| TRUE | OR | FALSE | FALSE |
| FALSE | OR | FALSE | FALSE |
| | NOT | TRUE | FALSE |
| | NOT | FALSE | TRUE |

To run in the boys 100m midget race at the school carnival you must be a boy and be in the midget age group then :

- If you are a boy but you are overage, then you cannot compete in the midget 100m.
- If you are in the right age group but are not a boy then you cannot compete in the midget 100m.
- If you are a boy and you are the right age then you can compete in the midget 100m. (and Kava will still win the House Sports)

The logic unit uses statement commands such as AND, OR, and NOT. This is useful when you have a set of instructions to execute/follow only if certain conditions are true. For example:

If the bus comes after school AND stops to pick me up,
then I can go home by bus,
Otherwise, I will have to walk home after school.

If I get my assignment finished AND hand in on time,
then I will get marks for my work,
Otherwise, I will not get any marks at all.

If my teacher comes to class, or a supervisor comes
then I will stay in the classroom and copy the written work into my book.

If NOT my teacher comes to class or a supervisor comes,
then I will do what I feel like doing.

For computers, since the data is internally understands is 0 or 1 the comparison is more straight forward with Table 4.3 outlining the definitions agreed to by early programmers as to the answer for comparisons of 0 and 1.

## Memory Cache

As briefly outlined in the discussion on CPU, the CPU makes its computations by taking information from memory, manipulating it and returning certain values to memory.

The CPU itself contains cache memory. This internal memory is sufficient to make some calculations, but not sufficient to contain additional complex instructions such as the pictures you see on your video games and computers. This internal memory is usually referred to as the *cache*, a special type of memory designed to cache, or pre-fetch, some of the instructions that the CPU considers to be frequently used.

That is why on the IBM PC designed motherboard, we have “memory banks”, where extra memory for the CPU can be added to the machine.

Again, because the computer looks for things in lots of “2”, the general allocation of memory is in blocks of 4MB, 8MB etc. On newer machines the memory allocation is generally in blocks of 16MB, 32MB, etc.

Memory becomes important when the instructions, or program, that the CPU has to execute is larger than the amount of memory that the CPU has access to. In most cases, the CPU is then unable to execute the program and it is unclear how the CPU will behave. Today, the general saying “more is better” holds true for memory. If you can afford to get more, “go for it.”

Instructions to the CPU, as shown in our diagram, are usually the instructions from a computer program. The CPU will also want some memory space to store its calculations.

## Review Questions

1. Give two reasons why the binary number system is important to computers

2. Given that “A” has a decimal value of 65, encode the following in binary: (show all working)

a) C ______________________________

b) O ______________________________

c) P ______________________________

d) Y ______________________________

3. What do the following acronyms stand for ?

a) ASCII: ______________________________

b) BIT: ______________________________

c) BYTE: ______________________________

4. When dealing with memories, the term “WORD” usually pops up. Explain what a word is ?

5. Describe the relationship between a BIT, BYTE, ASCII and WORD.

6. The CPU contains 3 main parts, namely the Control Unit, ALU and memory (registers). List down the function of each part.

Control:
ALU:
Memory:

# Sources and References

Shelly & Cashman "Computer Fundamentals for an Information Age"
H.L. Capron "Computers, Tools for an Information Age." 2nd Edition.
Brian D. Powell "Computer Studies Revision"
Siliva, Tonga. Liahona High School 1997 Term 4 Final Exam, Computer Studies Form 5 (TSC)

http://www.tongatapu.net.to/compstud/ - Computer Studies Course Notes
http://www.tongatapu.net.to - **Tonga** on the **'NET**

http://www.tongatapu.net.to is available on all networked computers at Queen Salote College.