Topic: 1.3.2 Computer architecture and the fetch execute

Von Neumann Architecture

The earliest computing machines had fixed programs. For example, a desk calculator (in principle) is a fixed program computer. It can do basic mathematics, but it cannot be used as a word processor or a gaming console. Changing the program of a fixed-program machine requires re-wiring, re-structuring, or re-designing the machine. The earliest computers were not so much "programmed" as they were "designed". "Reprogramming", when it was possible at all, was a laborious process, starting with flowcharts and paper notes, followed by detailed engineering designs, and then the often-arduous process of physically re-wiring and re-building the machine. It could take three weeks to set up a program on ENIAC (a computer of 1940s) and get it working.

The phrase Von Neumann architecture derives from a paper written by computer scientist John von Neumann in 1945. This describes design architecture for an electronic digital computer with subdivisions of a central arithmetic part, a central control part, a memory to store both data and instructions, external storage, and input and output mechanisms. The meaning of the phrase has evolved to mean a stored-program computer. A stored-program digital computer is one that keeps its programmed instructions, as well as its data, in read-write, random-access memory (RAM). So John Von Neumann introduced the idea of the stored program. Previously data and programs were stored in separate memories. Von Neumann realized that data and programs are indistinguishable and can, therefore, use the same memory. On a large scale, the ability to treat instructions as data is what makes assemblers, compilers and other automated programming tools possible. One can "write programs which write programs". This led to the introduction of compilers which accepted high level language source code as input and produced binary code as output.

Topic: 1.3.2 Computer architecture and the fetch execute

The Von Neumann architecture uses a single processor which follows a linear sequence of *fetch-decode-execute*. In order to do this, the processor has to use some special registers, which are discrete memory locations with special purposes attached. These are:

Register	Meaning
PC	Program Counter
CIR	Current Instruction Register
MAR	Memory Address Register
MDR	Memory Data Register
IR/IX	Index Register
Accumulator	Holds results
Status Register	Process states such as whether a result is zero, positive/negative or resulted in overflow.

- The **program counter** keeps track of where to find the next instruction so that a copy of the instruction can be placed in the current instruction register. Sometimes the program counter is called the Sequence Control Register (SCR) as it controls the sequence in which instructions are executed.

- The **current instruction register** holds the instruction that is to be executed.

- The **memory address register** is used to hold the memory address that contains either the next piece of data or an instruction that is to be used.

- The **memory data register** acts like a buffer and holds anything that is copied from the memory ready for the processor to use it.

The central processor contains the **arithmetic-logic unit** (also known as the arithmetic unit) and the **control unit**. The arithmetic-logic unit (ALU) is where data is processed. This involves arithmetic and logical operations. Arithmetic operations are those that add and subtract numbers, and so on. Logical operations involve comparing binary patterns and making decisions.

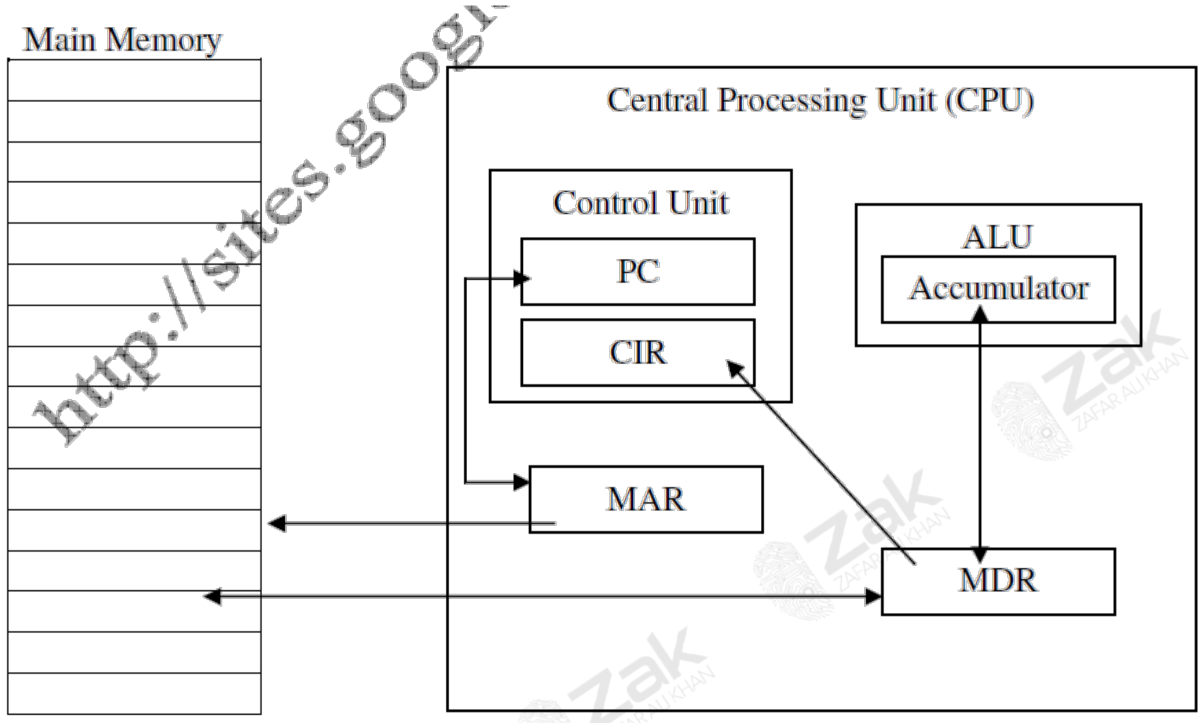
The control unit fetches instructions from memory, decodes them and synchronizes the operations before sending signals to other parts of the computer.

- The **accumulator** is in the arithmetic unit, the program counter and the instruction registers are in the control unit and the memory data register and memory address register are in the processor.

- An **index register** is a microprocessor register used for modifying operand addresses during the run of a program, typically for doing vector/array operations. Index registers are used for a special kind of indirect addressing where an immediate constant (i.e. which is part of the instruction itself) is added to the contents of the index register to form the address to the actual operand or data.

Topic: 1.3.2 Computer architecture and the fetch execute

A typical layout is shown in following which also shows the data paths.



Topic: 1.3.2 Computer architecture and the fetch execute

The Fetch-Decode-Execute-Reset Cycle

The following is an algorithm that shows the steps in the cycle. At the end, the cycle is reset and the algorithm repeated.

```
1. Load the address that is in the program counter (PC) into the memory
   address register (MAR).
2. Increment the PC by 1.
3. Load the instruction that is in the memory address given by the MAR into
   the memory data register (MDR).
4. Load the instruction that is now in the MDR into the current instruction
   register (CIR).
5. Decode the instruction that is in the CIR.
6. If  the instruction is a jump instruction then
      a. Load the address part of the instruction into the PC
      b. Reset by going to step 1.
7. Execute the instruction.
8. Reset by going to step 1.
```

Steps 1 to 4 are the *fetch* part of the cycle. Steps 5, 6a and 7 are the *execute* part of the cycle and steps 6b and 8 are the *reset* part.

Step 1 simply places the address of the next instruction into the **memory address register** so that the control unit can fetch the instruction from the right part of the memory. The program counter is then incremented by 1 so that it contains the address of the next instruction, assuming that the instructions are in consecutive locations.
The **memory data register** is used whenever anything is to go from the central processing unit to main memory, or vice versa. Thus the next instruction is copied from memory into the MDR and is then copied into the current instruction register.

Now that the instruction has been fetched the control unit can decode it and decide what has to be done. **This is the execute part of the cycle**. If it is an arithmetic instruction, this can be executed and the cycle can restart as the PC contains the address of the next instruction in order. However, if the instruction involves jumping to an instruction that is not the next one in order, the PC has to be loaded with the address of the instruction that is to be executed next. This address is in the address part of the current instruction, hence the address part is loaded into the PC before the cycle is reset and starts all over again.

A CPU cannot do math on data registers, although it can do it indirectly with an index register. The index register works with the data registers, allowing a program to process strings of data efficiently. To process your first name, for example, a program move 300 to MAR and zero to the index register. An indexed operation adds the index value to the MDR, retrieving the letter at location 300. Next, the program increments the index by one and gets the next letter. It repeats this process until it has moved the whole name. **By itself, the index register does little; its value is that it gives greater speed and convenience to address registers.**