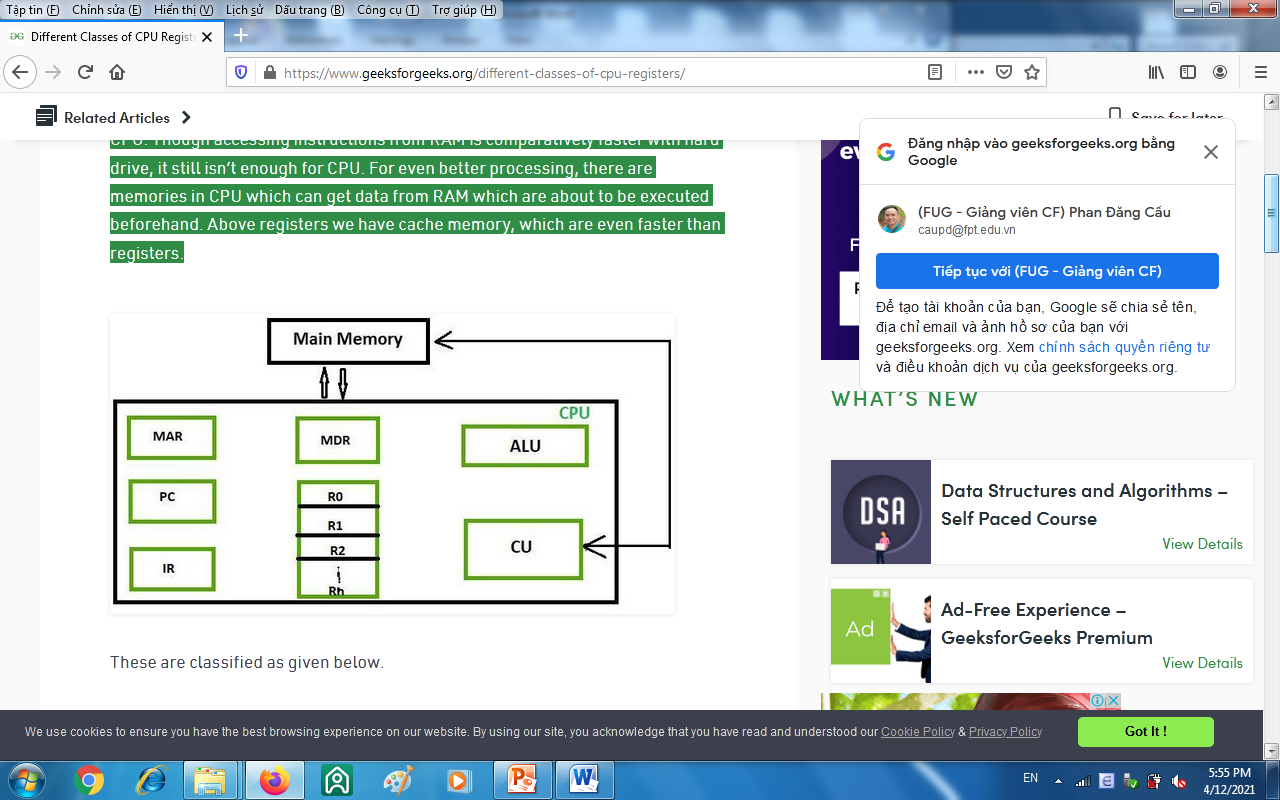
In Computer Architecture, the Registers are very fast computer memory which are used to execute programs and operations efficiently. This does by giving access to commonly used values, i.e., the values which are in the point of operation/execution at that time. So, for this purpose, there are several different classes of CPU registers which works in coordination with the computer memory to run operations efficiently.

The sole purpose of having register is fast retrieval of data for processing by CPU. Though accessing instructions from RAM is comparatively faster with hard drive, it still isn’t enough for CPU. For even better processing, there are memories in CPU which can get data from RAM which are about to be executed beforehand. Above registers we have cache memory, which are even faster than registers.



* **Accumulator:**   
  This is the most frequently used register used to store data taken from memory. It is in different numbers in different microprocessors.
* **Memory Address Registers (MAR):**   
  It holds the address of the location to be accessed from memory. MAR and MDR (Memory Data Register) together facilitate the communication of the CPU and the main memory.
* **Memory Data Registers (MDR):**   
  It contains data to be written into or to be read out from the addressed location.
* **General Purpose Registers:**   
  These are numbered as R0, R1, R2….Rn-1, and used to store temporary data during any ongoing operation. Its content can be accessed by assembly programming. Modern CPU architectures tends to use more GPR so that register-to-register addressing can be used more, which is comparatively faster than other [addressing modes](https://www.geeksforgeeks.org/addressing-modes/).
* **Program Counter (PC):**   
  Program Counter (PC) is used to keep the track of execution of the program. It contains the memory address of the next instruction to be fetched. PC points to the address of the next instruction to be fetched from the main memory when the previous instruction has been successfully completed. Program Counter (PC) also functions to count the number of instructions. The incrementation of PC depends on the type of architecture being used. If we are using 32-bit architecture, the PC gets incremented by 4 every time to fetch the next instruction.
* **Instruction Register (IR):**   
  The IR holds the instruction which is just about to be executed. The instruction from PC is fetched and stored in IR. As soon as the instruction in placed in IR, the CPU starts executing the instruction and the PC points to the next instruction to be executed.
* **Condition code register ( CCR ) :** Condition code registers contain different flags that indicate the status of any operation.for instance lets suppose an operation caused creation of a negative result or zero, then these flags are set high accordingly.and the flags are

1. Carry C: Set to 1 if an add operation produces a carry or a subtract operation produces a borrow; otherwise cleared to 0.
2. Overflow V: Useful only during operations on signed integers.
3. Zero Z: Set to 1 if the result is 0, otherwise cleared to 0.
4. Negate N: Meaningful only in signed number operations. Set to 1 if a negative result is produced.
5. Extend X: Functions as a carry for multiple precision arithmetic operations.

          and these are generally decided by ALU.

So, these are the different registers which are operating for a specific purpose.

--------------------------------------------------------------------------------------------------------------------

**Kernel perspective:**

[Memory segmentation](http://en.wikipedia.org/wiki/Memory_segmentation) is the old way of accessing memory regions. All major operating systems including OSX, Linux, (from version 0.1) and Windows (from NT) are now using [paging](http://en.wikipedia.org/wiki/Paging) which is a better way (IMHO) of accessing memory.

Before paging, the segment registers were used like this

**Let's see some examples (286-386 era) :**

The 286 architecture introduced 4 segments: **CS** (code segment) **DS** (data segment) SS (stack segment) **ES** (extra segment) the 386 architecture introduced two new general segment registers **FS**, **GS**.