Topic: 1.4.2 The fetch-execute cycle

The process by which a computer:

1. Fetches a program instruction from its memory,
2. determines what instruction wants to do,
3. And execute those actions.

This cycle is repeated continuously by the central processing unit (CPU), from boot up to shut down. In modern computers this means completing the cycle billions of times in a second! Without it, nothing would be able to be calculated.

Registers involved

The circuits used in the CPU during the cycle are:

- **Program Counter (PC)** - an incrementing counter that keeps track of the memory address of which instruction is to be executed next...
- **Memory Address Register (MAR)** - the address in main memory that is currently being read or written
- **Memory Data Register (MDR)** - a two-way register that holds data fetched from memory (and data ready for the CPU to process) or data waiting to be stored in memory
- **Current Instruction register (CIR)** - a temporary buffer for the instruction that has just been fetched from memory
- **Control Unit (CU)** - decodes the program instruction in the CIR, selecting machine resources such as a data source register and a particular arithmetic operation, and coordinates activation of those resources
- **Arithmetic logic unit (ALU)** - performs mathematical and logical operations

Topic: 1.4.2 The fetch-execute cycle

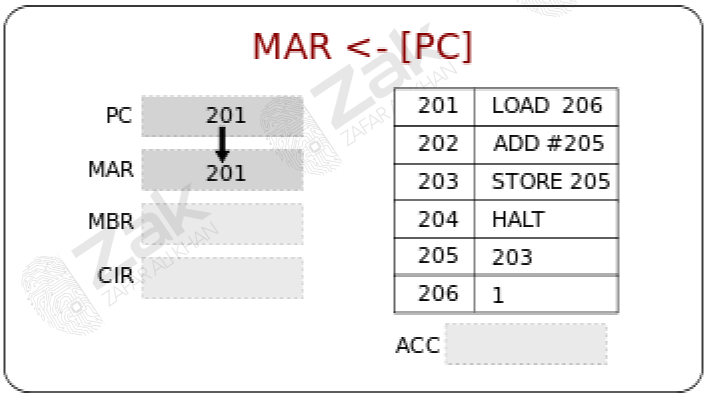
Register transfer notation

To describe the cycle we can use register notation. This is a very simple way of noting all the steps involved. In all cases where you see brackets e.g. [PC], this means that the contents of the thing inside the brackets are loaded. In the case of the first line, the contents of the program counter are loaded into the Memory Address Register.

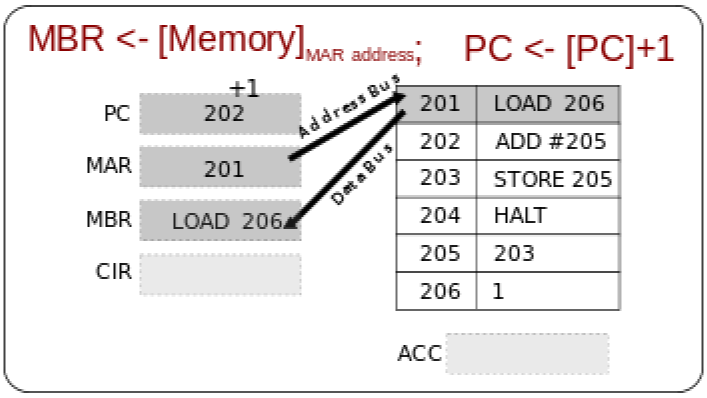
$MAR \leftarrow [PC]$
$MBR \leftarrow [Memory]_{MARaddress}; PC \leftarrow [PC] + 1$ (Increment the PC for next cycle at the same time)
$CIR \leftarrow [MBR]$ MBR is also called MDR
$[CIR]$ Decoded then executed

Detailed description of Fetch-Decode-Execute Cycle

To better understand what is going on at each stage we'll now look at a detailed description:

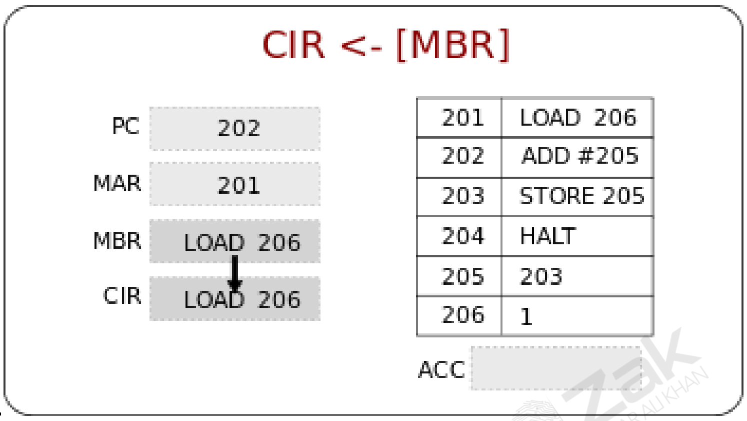


1. The contents of the Program Counter, the address of the next instruction to be executed, is placed into the Memory Address Register

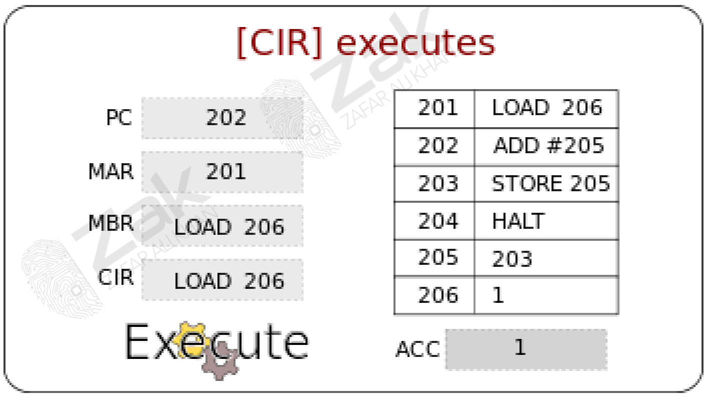


Topic: 1.4.2 The fetch-execute cycle

2. The address is sent from the MAR along the address bus to the Main Memory. The instruction at that address is found and returned along the data bus to the Memory Buffer Register also called Memory Data Register (MDR). At the same time the contents of the Program Counter is increased by 1, to reference the next instruction to be executed.



3. The MBR (MDR) loads the Current Instruction Register with the instruction to be executed.



4. The instruction is decoded and executed using the ALU if necessary.

The Cycle starts again!

Topic: 1.4.2 The fetch-execute cycle

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.

How Interrupts work:

1. Instruction Fetched
2. Current Inst. is executed
3. Interrupt presence is checked in queue
4. If there is an interrupt then check priority with current program
5. If Interrupt's priority is high then save current program's data in registers to STACK in RAM and load new program (Interrupt handler) by writing new program's 1st instruction address in PC.
    a. Serve the Interrupt
    b. Load the data back from STACK in RAM
    c. Resume previous program.
    6. Any more instruction of the current program?
    a. If yes then go to step 1.
    b. else end.

What happens in the CPU when an Interrupt is generated during a fetch execute cycle:
We have said that 'An interrupt is a signal for the CPU to stop what it is doing and instead carry out the interrupt task, once the task is complete, the CPU goes back to what it was doing'.
But what is meant by 'back to what it was doing'?
To appreciate this, you need to understand a little about what goes on inside a CPU. A CPU contains a number of 'registers'. A register is a small section of on-chip memory having a specific purpose. Registers range from 8 bits wide on an 8 bit CPU to 64 bits and beyond.
Registers in the CPU hold all of the data currently being handled. These include

- The current instruction being executed (Instruction Register),
- The location in primary memory of the next instruction (Program Counter)
- A number of general purpose registers holding current data

The registers are updated by each tick of the system clock so at any instant in time, they hold specific values. When an interrupt comes along, all the register values are copied to a special data structure or memory area called the 'stack' which is in primary memory. And they stay in the stack whilst the CPU starts executing the interrupt service routine (ISR). Once the routine is over, the registers are loaded back with their original values from the stack and can continue with what they were doing before the interrupt came along. This jumping of instructions from current CPU operations to the ISR and then back again is called 'context switching'