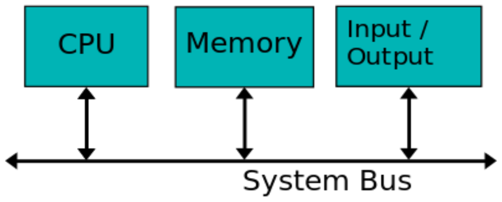
Unit 5 Processor Fundamentals

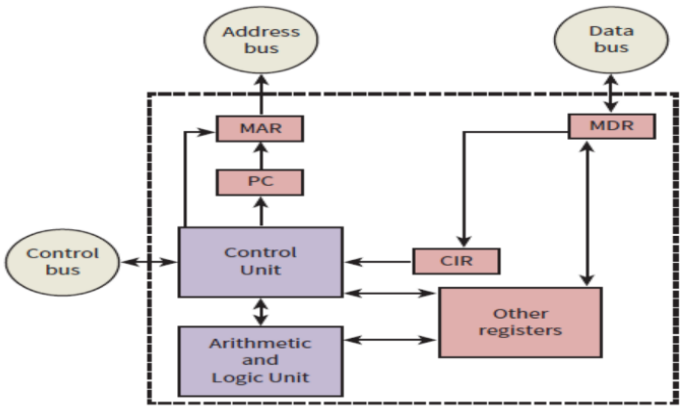
Von Neumann Machine

- All data and instructions are stored in the Main Memory.
- Instructions are sent to the Processor along the System Bus to be executed.
- Any input and output (such as printing and entering instruction) is performed by I/O devices with the data travelling from the I/O devices to the Processor and Main Memory by means of the System Bus:



Processor

The processor (or Central Processor Unit - CPU) is one of the most complex parts of any computer system. The processor executes programs and supervises the operation of the rest of the system. Single chip processors are otherwise known as microprocessors. Most modern processors contain billions of transistors. Multicore microprocessors are now very popular, where the processor will have several cores allowing for multiple programs or threads to be run at once.



A schematic diagram of the architecture of a simple CPU (Class Discussion)

Structure and role of the processor

Arithmetic logic unit

The Arithmetic Logic Unit or the ALU is a digital circuit that performs arithmetic and logical operations. Where arithmetic operations include things such as ADD and SUBTRACT and the logical operations include things such as AND, OR, NOT.

The ALU is a fundamental building block in the central processing unit (CPU) of a computer and without it the computer wouldn't be able to calculate anything! Some examples of assembly code instructions that would use the ALU are as follows (not all processors will have all these instructions):

ADD; add one number to another number
SUB; subtract one number to another number
INC; increment a number by 1
DEC; decrements a number by 1
MUL; multiply numbers together
OR; Boolean algebra function
AND; Boolean algebra function
NOT; Boolean algebra function
XOR; Boolean algebra function
JNZ; jump to another section of code if a number is not zero (used for loops and ifs)
JZ; jump to another section of code if a number is zero (used for loops and ifs)

Control unit

The control unit sits inside the CPU and coordinates the input and output devices of a computer system. It coordinates the fetching of program code from main memory to the CPU and directs the operation of the other processor components by providing timing and control signals.

Internal clock: controls the cycles of activity within the processor
System clock: controls the cycles of activity outside the processor.
Clock speed - The number of cycles that are performed by the CPU per second

The frequency defines the minimum period of time that separates successive activities within the system.

Exercise: Characteristics of a processor

Q: How many different addresses can an 8-line address bus address?

A: $256 = 2^8$

Q: How does the address bus affect main memory?

A: If you have a small address bus then you will be limited in the number of addresses you can talk to and therefore how much main memory you can directly address.

Q: How wide would the address bus have to be to talk to 1024 addresses?

A: 10 lines wide since 2^{10}=1024

Q: What is wrong with using a 9 bit address bus but having 700 memory locations in main memory?

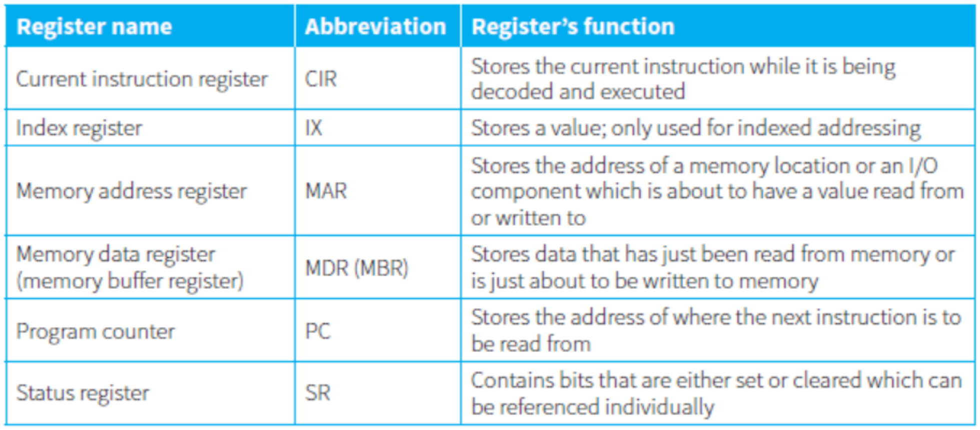
A: We can only address 2^{9}=512 different locations. It wouldn't be able to talk to address locations 513 to 700.

Registers:

The other components of the CPU are the registers. These are storage components which, because they are placed very close to the ALU, allow very short access times. Each register has limited storage capacity, typically 16, 32 or 64 bits. A register is either general purpose or special purpose.

Accumulator: A general-purpose register that stores a value before and after the execution of an instruction by the ALU.

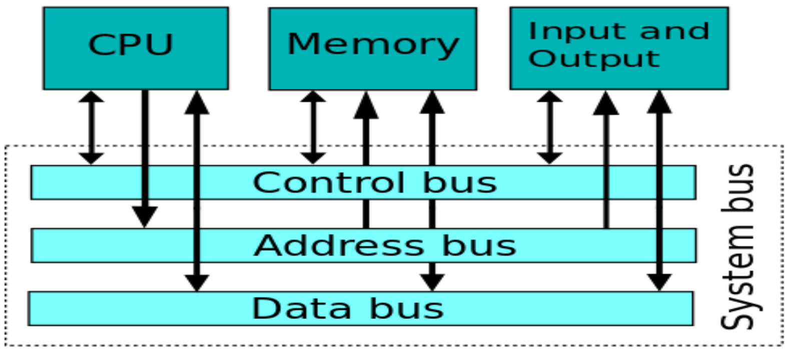
Special-Purpose Registers:



Register name	Abbreviation	Register's function
Current instruction register	CIR	Stores the current instruction while it is being decoded and executed
Index register	IX	Stores a value; only used for indexed addressing
Memory address register	MAR	Stores the address of a memory location or an I/O component which is about to have a value read from or written to
Memory data register (memory buffer register)	MDR (MBR)	Stores data that has just been read from memory or is just about to be written to memory
Program counter	PC	Stores the address of where the next instruction is to be read from
Status register	SR	Contains bits that are either set or cleared which can be referenced individually

The system Bus:

A bus is a parallel transmission component with each separate wire carrying a single bit. It is important not to describe a bus as a storage device. A bus does not hold data. Instead it is a mechanism for data to be transferred from one system component to another.



Address Bus

A **single-directional** bus that carries address signals from the CPU to Main Memory and I/O devices. This might involve the CPU requesting some data from Main Memory, sending the address of the data to Main Memory, then Main Memory returning the data along the data bus. Let's take a look at some code:

```
LDA 23
```

This code is asking to load the data from memory address 23 into the CPU, the address bus does not send addresses to the processor, but only sends them from the processor. To do this the CPU send 23 along the Address Bus, and the value from memory location 23 would be sent along the Data Bus back to the CPU.

Data bus

A **bi-directional** bus, typically consisting of 32 wires, used to transport data and instructions between the three components of the three-box model. The larger the Data Bus the more data can be transported at one time. For example, if we have an 8-bit Data Bus, the maximum value we could send along the Bus would be:

1111 1111 = 255

The larger the Data Bus the more data we can send at once and the more complex instructions we can use.

Control bus

A **bi-directional bus**, typically consisting of more than 16 wires, used to transport control signals between the three components of the three-box model. The control bus is used to carry important information such as messages to say when a device has finished a job or when a device has just been plugged in. A simple example would be when you plug in your USB key and after a few moments a screen pops up asking you what you want to do with it. The control bus also contains interrupt signals which allow devices (printers, keyboards, disks, etc.) to signal that they have finished a request. The CPU temporarily suspends its current program, services the device and then resumes the previous program.

Increasing Performance:

If we want to increase the performance of our computer, we can try several things

- Increasing the clock speed
- Adjusting word length
- Increasing bus widths

But what is to stop us increasing the clock speed as much as we want? If you study Physics you might already know this, but the problem with increased clock speed is that an increased current will have to flow through the circuits. The more current that flows, the hotter things get. You might notice that a laptop will get hot or even your mobile phone when you are doing something processor intensive like playing a game. The faster the clock speed, the hotter the processor runs. To counter this computer scientists have come up with smarter chip designs and introduced heat sinks, fans, and even liquid cooling into computers. If a processor runs too hot it can burn out!

NES processor	**Heat sink**	**CPU fan**	**Water cooling**
No need for fans or heat sinks	Draws the heat away from the processor, which sits beneath	I love the CPU so much, I'm a real CPU fan	Metal contacts are placed on the CPU drawing heat away, water then passes over these contacts to draw heat away

Word Size - The number of bits of information that a processor can process at one time

Another way to increase the performance of a computer is to increase the word size. Modern computer mostly has 32- or 64-bit word sizes, with specialist hardware such as games consoles being able to handle up to 128-bit words.

Bus Size - The number of bits of information a bus can carry at one time (the number of wires making up a bus)

Exercise:

Q: What draw back might increasing clock speed bring?

A: Processor might need extra cooling hardware to stop it over heating

Q: What is a benefit of increasing word length?

A: Computers can perform more complex instructions in one go, dealing with larger numbers and greater number accuracy

Q: How might bus width impact on the speed of a computer?

A: If the bus width is smaller than the word size, then the CPU will have to wait around whilst the bus delivers data and instructions to it.

Peripherals:

Input/output devices are used by the system to get information in and out, as they are not internal but are connected to the CPU, we refer to them as **peripherals** (your hands are peripheral to your torso). We'll cover the specific ones you need to learn a little later, but for the moment you need to know the fundamental difference:

- Input Devices - used to get information into the system. E.g. Keyboard
- Output Devices - used to send information out of the system. E.g. Visual Display Unit (VDU)

If you look at the Von Neumann Architecture notice that it doesn't mention Keyboard or display, this is a very smart move as you don't want to force every computer to have a keyboard (think about a games console, there is no keyboard on that) or a VDU (some computer such as MP3 players don't have a screen). However, some computer architecture does include specific I/O controllers:

I/O Controllers:

An I/O controller is an electronic circuit that connects to a system bus and an I/O device; it provides the correct voltages and currents for the system bus and the I/O device. Examples would include:

- keyboard controller, attached to a keyboard
- disk controller for a Hard Disk (Hard Disks are not main memory!)
- video display controller, attaching a video display unit (monitor)

I/O Ports:

I/O ports is a complementary method of performing input/output between the CPU and peripheral devices in a computer. This allows I/O devices to be connected to the CPU without having to have specialist hardware for each one. Think about the USB port on your computer, you can connect Keyboards, Mice, Game pads, Cameras, Phones, etc. and they all connect using the same port.

The Universal Serial Bus (USB): It uses Asynchronous serial data transmission. Don't forget that the USB is a bus. A USB drive stores data and is connected to a USB port which allows data to be transmitted along the bus.

The following is some information about the USB standard.

- A hierarchy of connections is supported.
- The computer is at the root of this hierarchy and can handle 127 attached devices.
- Devices can be attached while the computer is switched on and are automatically configured for use.
- The standard has evolved, with USB 3.2 being the latest version.

Specialized Multimedia Ports:

Despite the widespread use of USB ports there are some peripheral devices that require a different port, one that is specialized for the type of device. Although computer systems come packaged with a monitor for screen display there is sometimes a requirement for a second screen to be used. The connection of the second screen can be through a Video Graphics Array (VGA) port. This provides high-resolution screen display which is suitable for most display requirements. However, if the screen is needed to display a video, the VGA port is not suitable because it does not transmit the audio component. A High Definition Multimedia Interface (HDMI) port will provide a connection to a screen and allow the transmission of high-quality video including the audio component.

Stored Program Concept - a program must be in main memory in order for it to be executed. The instructions are fetched, decoded and executed one at a time

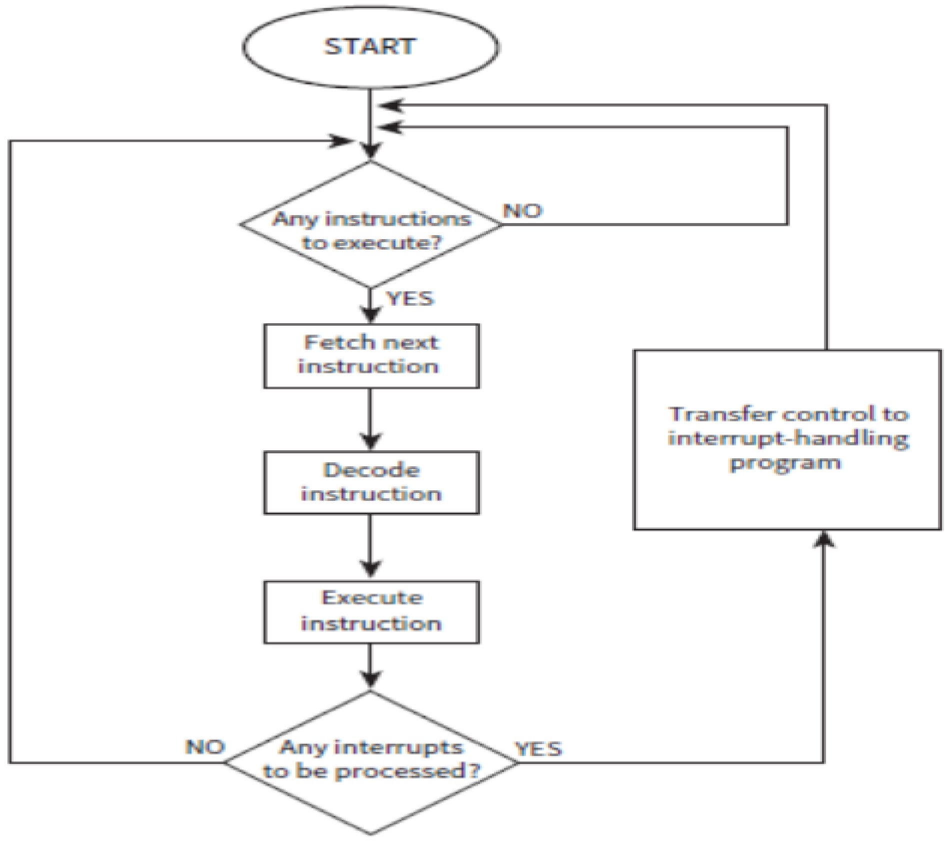
Building on the Von Neumann architecture we get the idea of how the stored program concept works. If you have ever loaded a game on a console you might notice that:

1. you need to insert a disc
2. the disc spins
3. the game says loading
4. the game plays

This is the stored program concept in motion! Let's take apart what is happening:

1. You insert an optical disk (secondary storage) with the code on
2. The code is loaded into main memory
3. The processor fetches, decodes and executes instructions from main memory to play game.

The Fetch–Execute (F–E) Cycle:


START
Any instructions to execute?
NO
YES
Fetch next instruction
Decode instruction
Execute instruction
Any interrupts to be processed?
NO
YES
Transfer control to interrupt-handling program

If we assume that a program is already running, then the program counter will already hold the address of the next instruction. In the fetch stage, the following steps will now happen.

1 This address in the program counter (PC) is transferred within the CPU to the MAR.
2 During the next clock cycle two things happen simultaneously:

- The instruction held in the address pointed to by the MAR is fetched into the MDR.
- The address stored in the program counter is incremented.

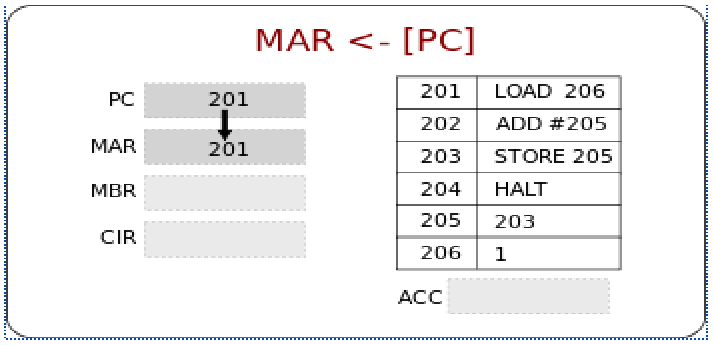
3 The instruction stored in the MDR is transferred within the CPU to the CIR.
There are two points to note here.

- The clock cycle is the one controlled by the system clock which will have settings that allow one data transfer from memory to take place in the time defined for one cycle.
- In the final step the program counter is incremented by 1.

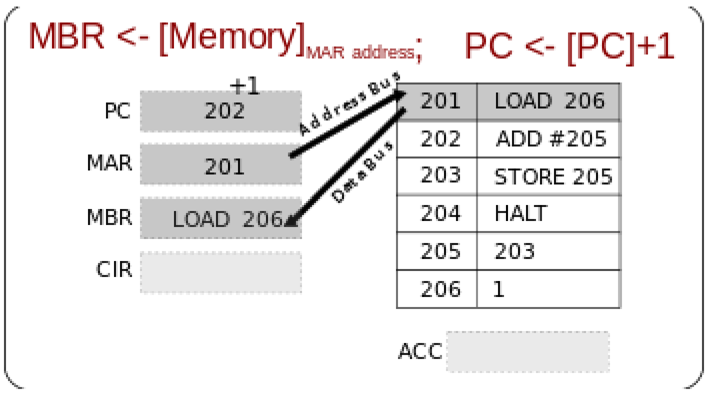
In the decode stage, the instruction stored in the CIR is received as input by the circuitry within the control unit. Depending on the type of instruction, the control unit will send signals to the appropriate components so that the execute stage can begin. At this stage, the ALU will be activated if the instruction requires arithmetic or logic processing.

Detailed description of Fetch-Decode-Execute Cycle

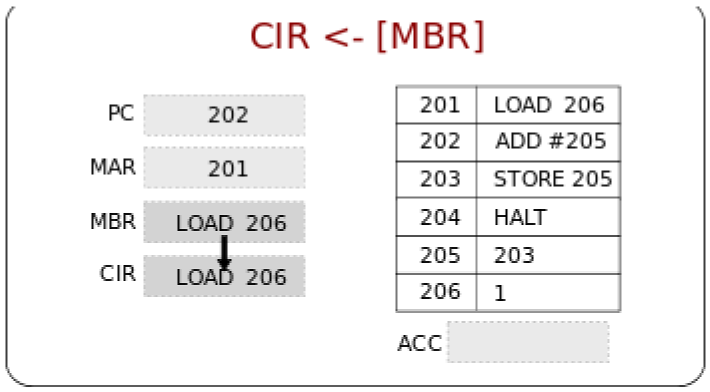
To better understand what is going on at each stage we'll now look at a detailed description:



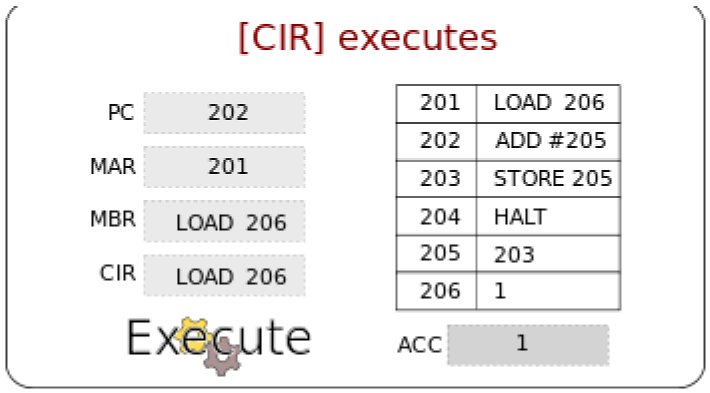
The contents of the Program Counter, the address of the next instruction to be executed, is placed into the Memory Address Register.



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. At the same time the contents of the Program Counter is increased by 1, to reference the next instruction to be executed.



The MBR loads the Current Instruction Register with the instruction to be executed.



The instruction is decoded and executed using the ALU if necessary. The Cycle starts again!

Interrupt Handling:

There are many different reasons for an interrupt to be generated. Some examples are:

- a fatal error in a program.
- a hardware fault.
- a need for I/O processing to begin.
- user interaction.
- a timer signal.

1. The contents of the program counter and any other registers are stored somewhere safe in memory.
2. The appropriate interrupt handler or Interrupt Service Routine (ISR) program is initiated by loading its start address into the program counter.

3. When the ISR program has been executed there is an immediate check to see if further interrupts need handling.
4. Further interrupts are dealt with by repeated execution of the ISR program.
5. If there are no further interrupts, the safely stored contents of the registers are restored to the CPU and the originally running program is resumed.