Microcontroller Architecture

Micro Architecture

Micro architecture refers to the main circuits in a MPU, how they are arranged, and how they communicate with one another and the outside world. The main circuits are: registers, arithmetic logic unit (ALU), control circuits, memory reference circuits, and input/output (I/O) circuits

Architecture also refers to the type and number of internal and external communications buses.

The number and type of instructions that can be executed further defines the architecture.

The Importance of Architecture

The architecture determines the micro performance.

Performance is stated in terms of processor clock frequency.

Another measure of performance is the instruction execution speed. It is generally measured in millions of instructions (executed) per second (MIPS).

Memory access speed is another measure of performance.

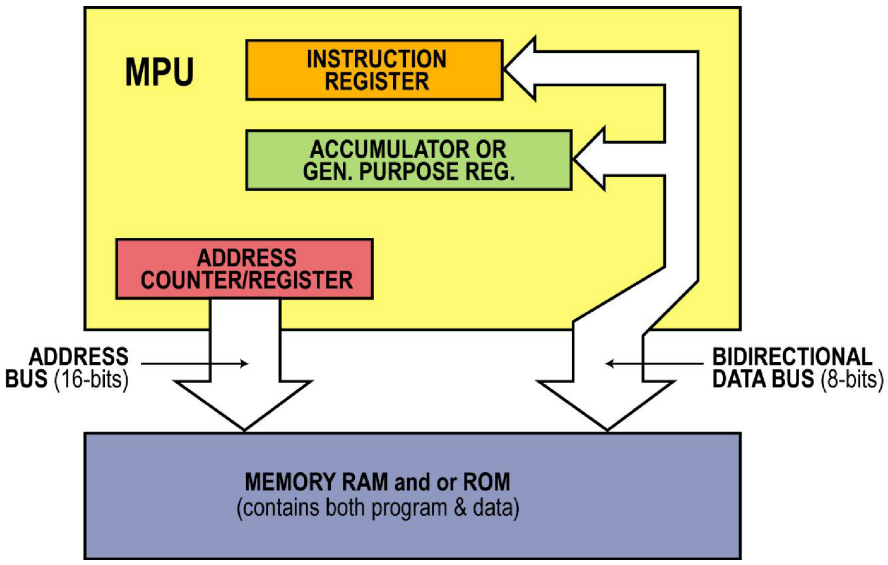
Von Neumann Architecture

The most common form of micro architecture is the Von Neumann form.

John Von Neumann was the physicist who invented the stored program concept and basic digital computer architecture in the late 1940's.

The stored program concept means that the operation of the computer is controlled by a sequence of binary instructions called a program that is stored in memory. The instructions are fetched, decoded, and executed one at a time in sequence.

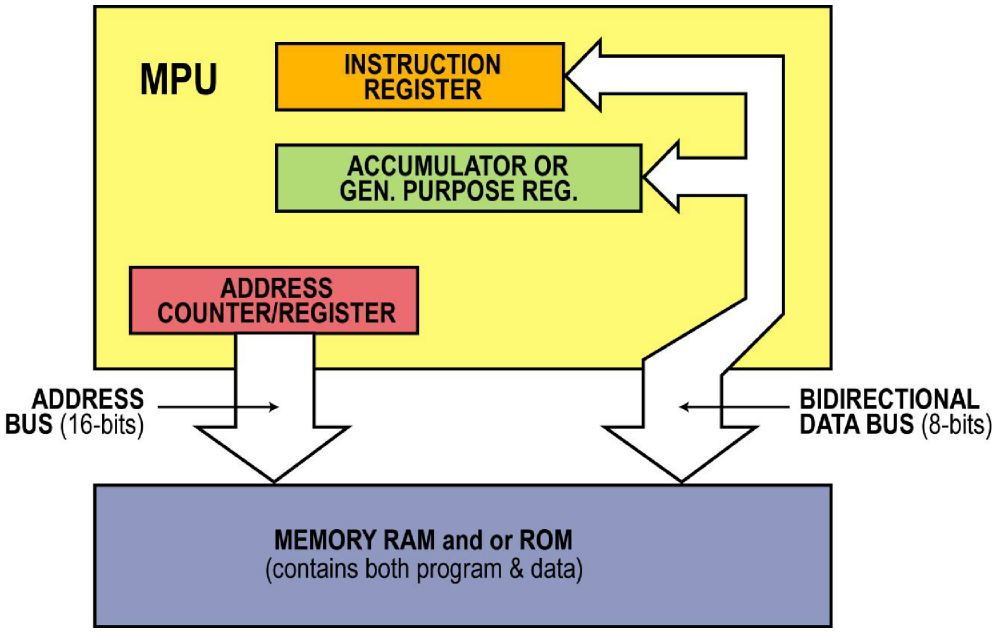
Von Neumann Architecture Operation



In a Von Neumann architecture CPU, both the binary instructions and data to be processed are stored in the same memory space.

This memory space is defined by an address counter/register that determines where the program and data are stored and how they are accessed. The output of this address counter forms the address bus.

Von Neumann Architecture Instructions



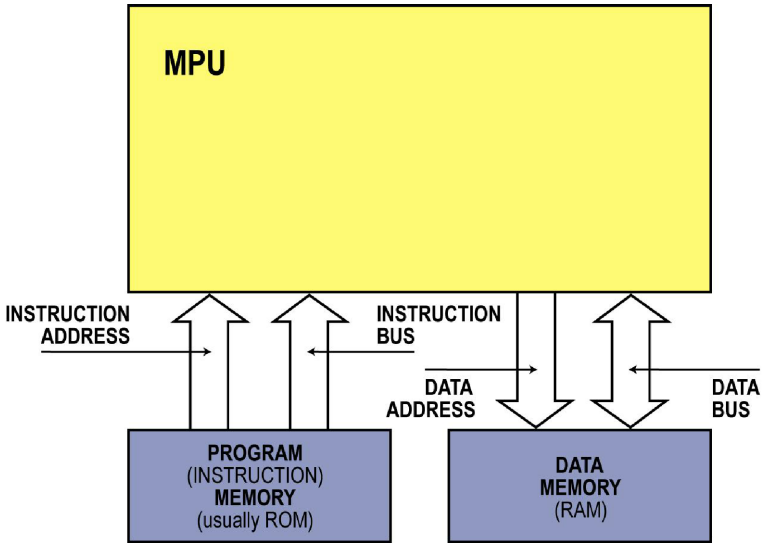
All read and write accesses to memory take place over a single data bus. Instructions to be executed are accessed then passed to the MPU over the data bus. Then the data to be processed is accessed via the same bus. The memory location to be accessed is identified by a binary address number.

Von Neumann Architecture Limitations

The Von Neumann architecture works well but it limits performance because both instructions and data must be accessed over the same data bus one after the other. This architectural feature limits CPU performance and is generally referred to as the “Von Neumann bottleneck”.

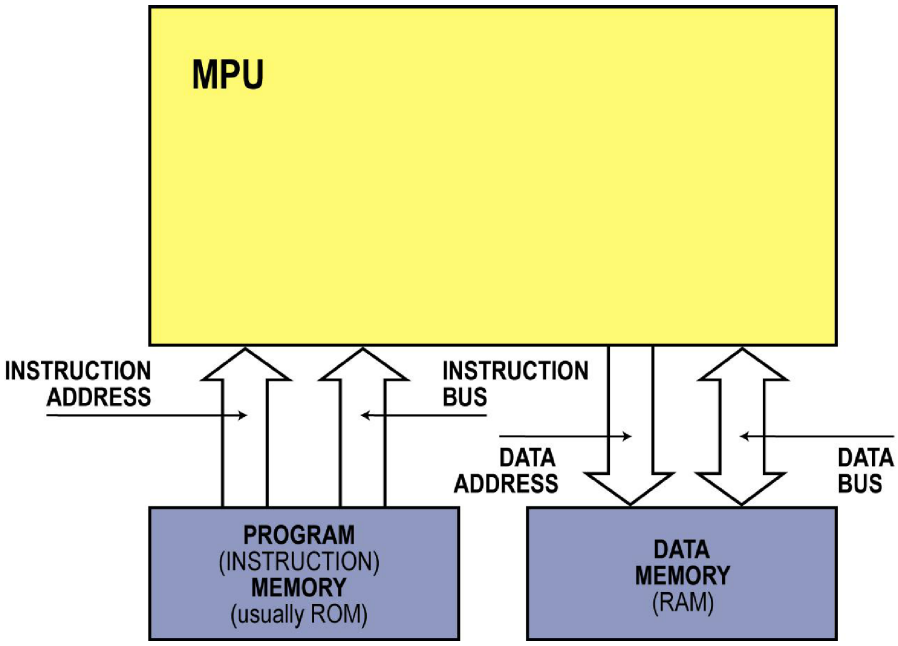
Modern CPUs using this architecture work well simply because the very high speeds at which the circuitry runs overcomes this natural limitation.

Harvard Architecture



To improve program execution and data access speed a newer form of architecture was developed. Known as the Harvard architecture, this format uses separate program and data memories with separate address counter/registers and buses. Separate program and data memories speed program execution by allowing simultaneous instruction fetch and data access thereby accelerating all parts of the instruction fetch- execute cycle.

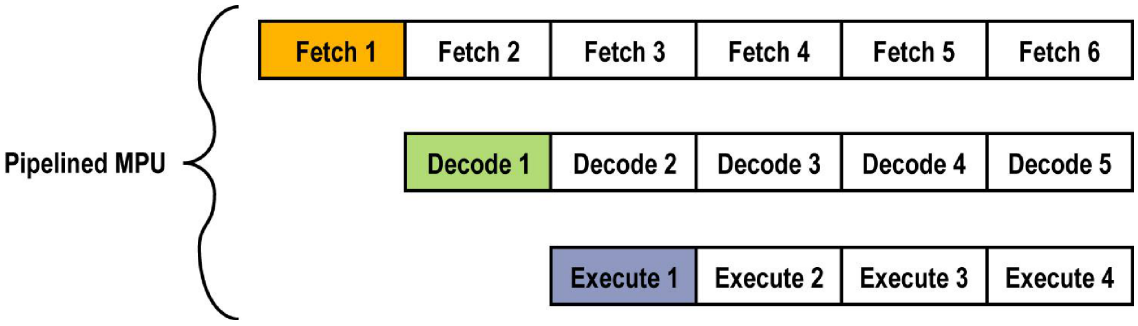
Harvard Architecture Implementation



Harvard architecture is somewhat more complex in implementation but with modern IC techniques it is not a limitation.

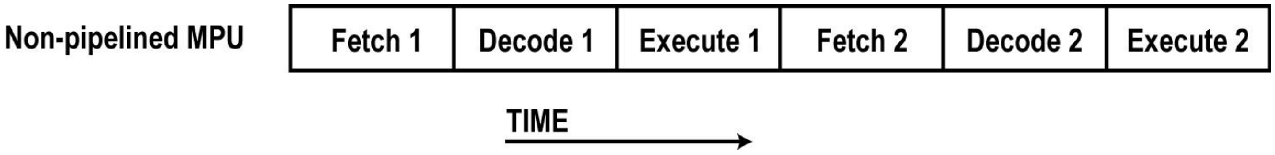
Most Harvard architecture micros use ROM or flash memory for the program memory and static read/write RAM for the data memory.

Pipelining

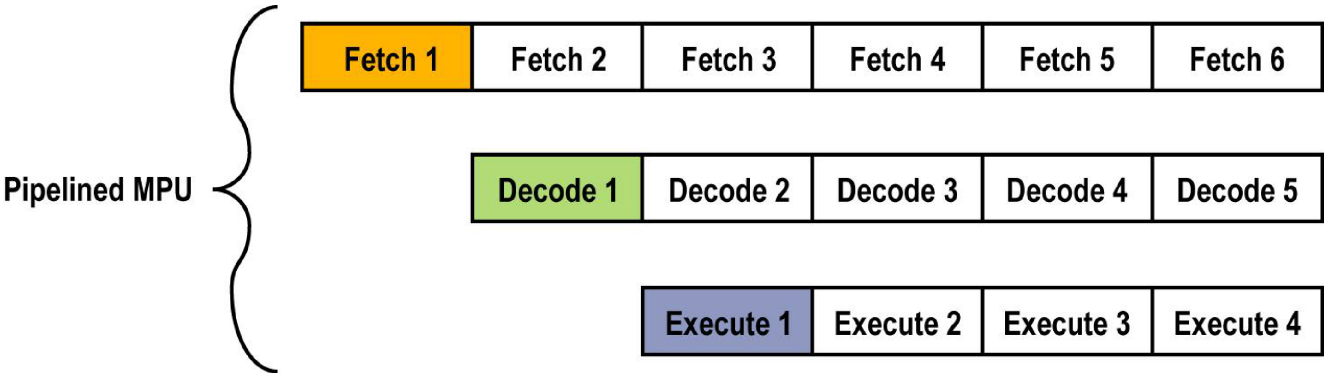


Another architectural feature of the modern micro is pipelining. Pipelining (shown above) is the technique of allowing multiple parallel fetch, instruction decode, and execute cycles.

In a standard Von Neumann or Harvard architecture micro, instructions are fetched one at a time from memory then decoded and executed as shown in the figure below.



Pipelined Architecture Operation



In pipelined architecture, circuitry is provided so that as one instruction is fetched, decoded and executed, the next instruction in sequence can be fetched during the same time as the decode of the first instruction.

Multiple pipelines may be implemented in circuitry to permit two, three, or more concurrent fetch-execute streams that significantly increase processing speed.

Other Forms of Architecture

There are two other forms of architectures to which micros conform. They are complex instruction set computing (CISC) and reduced instruction set computing (RISC)

Original computers and micros were of the CISC design.

Newer micros and computers use the RISC format although some newer micros are a mix of RISC and CISC.

Complex Instruction Set Computing

One way to make computers more powerful is to add more instructions.

In CISC designs, additional logic circuitry is added for each instruction to be implemented.

Remember that the instructions define the operations the CPU can perform including data movement, memory access, arithmetic, logic, or I/O.

Older computers only performed basic data movement tasks and arithmetic operations. More complex operations were programmed with the simple instructions.

Pros and Cons of CISC

CISC micros have large instruction sets and generally make the CPU faster and easier to program.

An example can be seen by looking at older simpler computers. The process of multiplication was accomplished by a program or subroutine that used addition and shifting operations. This program had to be written.

In newer CISC CPUs, a multiply instruction is provided to implement this function without programming. Multiply logic circuitry carries out the instruction. No programming is required.

Accessing Data

Most of the additional instructions in a CISC CPU are memory reference instructions that cause the circuitry to access data in memory one or more times during the execution. Retrieving data from memory is a time consuming process because memories are inherently slower than CPU circuits.

The primary downside of this feature is larger, more complex circuits, and in some cases, decreased execution speeds and higher power consumption.

RISC Architecture

Reduced instruction set computing (RISC) is an architecture that minimizes the total number of instructions a CPU can execute. This decreases the circuit quantity and complexity while increasing execution speed.

RISC architecture substantially improves overall computing performance in specific types of applications.

Key RISC Features

A primary feature of RISC CPUs is the use of only a few memory access instructions. These are usually load or store instructions that are used to load a register from memory or store a register content to memory.

Memory access operations are slow because most memory circuits are slower than CPU circuits. The fewer references to memory the faster the program execution.

Most RISC CPUs use Harvard architecture and are pipelined.

RISC Operation

RISC CPUs are also characterized by a large bank of registers called general purpose registers (GPR) that are used in conjunction with other instructions. The approach used in RISC is for instructions to operate on data in the registers. Such operations can take place significantly faster between registers than they can if memory accesses were involved.

RISC architecture also limits the total number of instructions thereby greatly simplifying circuitry and making the circuits simpler and faster. RISC chips also draw less power.

Some RISC CPUs may not even have a multiply instruction so multiply operations have to be programmed. However, the multiply operation could execute faster than a multiply instruction in a CISC CPU because of the RISC architectural features.

CISC vs. RISC

Both CISC and RISC CPUs have their place. For example, the Pentium and Opteron used in PCs are CISC designs with hundreds of instructions. But with modern IC technology, performance is impressive with clock speeds in excess of 3 GHz. Dual cores make them even faster.

However, while these processors are fine for general purpose computing where the applications vary widely, there are some operations that do not require the flexibility of a large instructions set. Networking and telecommunications are examples. Such applications are better suited to the sparse but super fast execution of RISC MPUs.

Some of the newer processors actually are hybrids combining some features of both RISC and CISC to optimize their use in specific applications.

Fixed vs. Floating Point Math

Most micros work with fixed length numbers (8, 16, 32, 64-bits).

The size of the number dictates the largest and smallest numbers that can be represented and processed.

For example, even with a 32-bit word, the maximum value is 2^{32}= 4,294,967,296 or just over 4 billion. That may be insufficient for some applications.

Floating point capability greatly expands the number size and dynamic range.

Floating Point Numbers

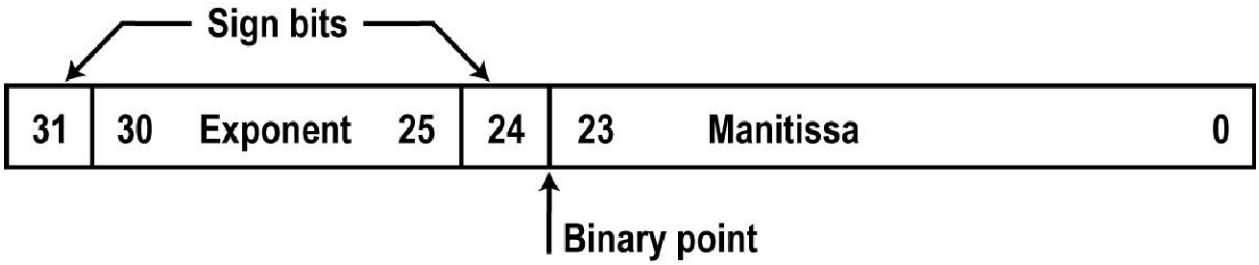
Floating point is the computer binary equivalent of scientific notation where a very large or small quantity is represented by a number between 1 and 10 multiplied by some power of 10.

$65{,}000{,}000{,}000 = 6.5 \times 10^{9}$

$0.00000000000038 = 3.8 \times 10^{-13}$

Floating point uses a similar technique where any value is represented by a binary value between 0.1 and 1 multiplied by a power of 2.
Example: $.101 \times 2^{-3} = .000101 = .078125$

Floating Point Word Format



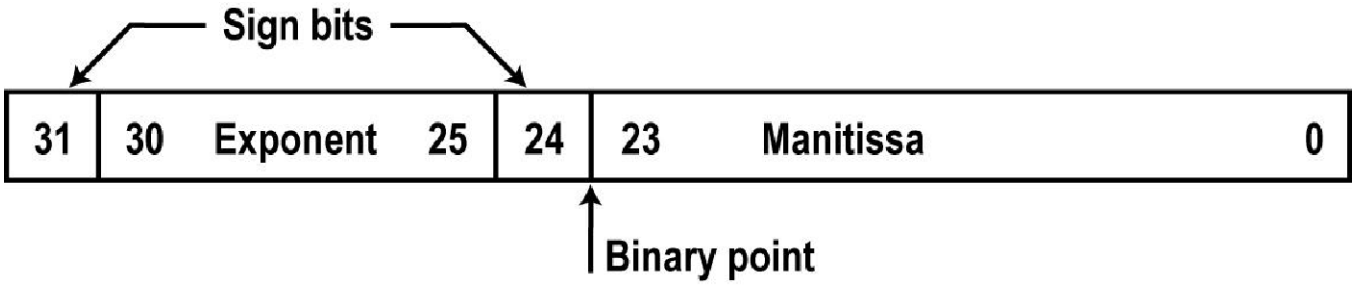
The figure above shows a 32-bit floating point data word. Some computers use an 80-bit floating point word.

The 24-bit (bits 0 – 23) fixed fraction is called the mantissa. Bit 24 is a sign bit.

The power of 2 is the 8-bit exponent (bits 25 - 31).

With this format, values from 2^{-64} to 2^{63} can be represented.

Floating Point Math Logic Circuitry



Floating point words require special math logic circuitry. It is slower than standard fixed point math circuits.

Floating point math instructions (add, multiply, etc.) allow very large and very small numbers to be manipulated.

Most applications do not require floating point; however, many scientific, engineering, and DSP applications do require floating point.

Test your knowledge

Micro & Embedded Controllers
Part 1: Microcontroller Technology Update
Knowledge Probe 2
Microcontroller Architecture

Click on Course Materials at the top of the page.

Then choose **Knowledge Probe 2**.