**CONTENTS**

1. INTRODUCTION

* SPECIFICATION

1. PROCESSOR AND ARCHITECHTURE
2. INSTRUCTION SET
3. MEMORY ORGANIZATION
4. CONTROL UNIT
5. I/O MECHANISMS
6. LEGAL INFORMATION

**INTRODUCTION**

IdeaPad series was announced in January 2008. The first three models in product line were the Y710, the Y510, and the U. Some of the features that defined these first three models were widescreens , VeriFace facial recognition, frameless screens, touch controls, and Dolby speaker systems.

The IdeaPad design marked a deviation from the Business oriented ThinkPad laptops, towards a more consumers -oriented look and feel.

Among these changes were a glossy screen and the absence of the traditional ThinkPad TrackPoint. Notebook Review said the Keyboard had a “distinctive ThinkPad feel” and “the touchpad and touchpad button were smooth and responsive”.

On September 21, 2016, Lenovo confirmed that Yoga series is not meant to be compatible with Linux Operating systems that they know it is not supported. This came in wake of media coverage of problems that users were having while trying to install Ubuntu on several yoga models, including the

900,ISK2, 900 ISK For Business,900S and 710, which were traced back to Lenovo intentionally disabling and removing support for the AHCI storage mode for the device’s solid-state drive in the computer’s BIOS, in favor of a RAID mode that is only supported by windows 10 drivers that come with system

* **SPECIFICATIONS**

* Processor : AMD A6
* ROM : 500GB
* Installed Memory(RAM) : 4.00GB(3.81 GB usable)
* System Type : 64-bit operating system, x64-based processor
* Pen and touch : Touch support with 2 POINTS

This Laptop has a Slim Body with 180 degree open-fold feature.



**Processor and Architecture**

This Laptop have “AMD E2-9000 RADEON R2, 4 COMPUTE CORE 2C+2G 1.80GHzs” processor with 64-bits operating system.

AMD stands for Advanced Micro Design. Advanced Micro Devices, Inc. (AMD) is an American multinational semiconductor company based in Santa Clara, California that develops computer processors and related technologies for business and consumer markets. While initially it manufactured its own processors, the company later outsourced its manufacturing, a practice known as fabless, after Global Foundries was spun off in 2009 AMD's main products include microprocessors, motherboard chipsets, embedded processors and graphics processors for servers, workstations, personal computers and embedded system application. AMD is the second-largest supplier and only significant rival to Intel in the market for x86-based microprocessors. Since acquiring ATI in 2006 AMD and its competitor Nvidia have maintained a duopoly in the discrete graphics processing unit (GPU) market. In 2008 the ATI division of AMD released the Tera Scale microarchitecture implementing a unified shader model. This design replaced the previous fixed-function hardware of previous graphics cards with multipurpose, programmable shaders. Initially released as part of the GPU for the Xbox 360, this technology would go on to be used in Radeon branded HD 2000 parts. Three generations of Tera Scale would be designed and used in parts from 2008-2014 In a 2009 restructuring, AMD merged the CPU and GPU divisions to support the companies APU's which fused both graphics and general purpose processing. In 2011, AMD released the successor to Tera Scale, Graphics Core Next (GCN). This new microarchitecture emphasized GPU compute capability in addition to graphics processing, with a particular aim of supporting heterogeneous computing on AMD's APUs. GCN's reduced instruction set ISA allowed for significantly increased compute capability over Tera Scale's very long instruction word ISA. Since GCN's introduction with the HD 7970 five generations of the GCN architecture have been produced from 2008 through at least 2017.   
  
Radeon Technology   
  
In September 2015, AMD separated the graphics technology division of the company into an independent internal unit called the Radeon Technology Group (RTG) headed by Raja Koduri. This gave the graphics division of AMD autonomy in product design and marketing. The RTG then went on to create and release the Polaris and Vega micro architectures released in 2016 and 2017, respectively. In particular the Vega, or 5th generation GCN, microarchitecture includes a number of major revisions to improve performance and compute capabilities. AMD motherboard chipsets Edit

Before the launch of Athlon 64 processors in 2003 AMD designed chipsets for their processors spanning the K6 and K7 processor generations. The chipsets include the AMD-640, AMD-751 and the AMD-761chipsets. The situation changed in 2003 with the release of Athlon 64 processors, and AMD chose not to further design its own chipsets for its desktop processors while opening the desktop platform to allow other firms to design chipsets. This was the "Open Platform Management Architecture" with ATI, VIA and SiS developing their own chipset for Athlon 64processors and later Athlon 64 X2 and Athlon 64 FX processors, including the Quad FX platform chipset from Nvidia.

In computer architecture, 64-bit integers, memory addresses, or other data units are those that are 64 bits (8 octets) wide. Also, 64-bit CPU and ALU architectures are those that are based on registers, address buses, or data buses of that size. 64-bit microcomputers are computers in which 64-bit microprocessors are the norm. From the software perspective, 64-bit computing means the use of code with 64-bit virtual memory addresses. However, not all 64-bit instruction sets support full 64-bit virtual memory addresses; x86x64 and ARMv8, for example, support only 48 bits of virtual address, with the remaining 16 bits of the virtual address required to be all 0's or all 1's, and several 64-bit instruction sets support fewer than 64 bits of physical memory address.  
  
The term 64-bit describes a generation of computers in which 64-bit processors are the norm. 64 bits is a word size that defines certain classes of computer architecture, buses, memory, and CPUs and, by extension, the software that runs on them. 64-bit CPUs have been used in supercomputers since the 1970s (Cray-1, 1975) and in reduced instruction set computing (RISC) based workstations and servers since the early 1990s, notably the MIPS R400, R8000, and R10000, the DEC Alpha, the Sun UltraSPARC, and the IBM RS64 and POWER3 and later POWER microprocessors. In 2003, 64-bit CPUs were introduced to the (formerly 32-bit) mainstream personal computer market in the form of x86-64 processors and the PowerPC G5, and were introduced in 2012 [1] into the ARM architecture targeting smartphones and tablet computers, first sold on September 20, 2013, in the iPhone 5S powered by the ARMv8-A Apple A7 system on a chip (SoC).  
  
A 64-bit register can hold any of 264 (over 18 quintillion or 1.8×1019) different values. The range of integer values that can be stored in 64 bits depends on the integer representation used. With the two most common representations, the range is 0-9,223,372,036,854,775,808 (−263) through 18,446,744,073,709,551,615 (264 − 1) for representation as an (unsigned) binary number, and 9,223,372,036,854,775,807 (263− 1) for representation as two's complement. Hence, a processor with 64-bit memory addresses can directly access 264 bytes (=16 exabytes) of byte-addressable memory.  
  
With no further qualification, a 64-bit computer architecture generally has integer and addressing processor registers that are 64 bits wide, allowing direct support for 64-bit data types and addresses. However, a CPU might have external data buses or address buses with different sizes from the registers, even larger (the 32-bit Pentium had a 64-bit data bus, for instance.[2] The term may also refer to the size of low-level data types, such as 64-bit floating-point numbers.

**INSTRUCTION SET**

An **instruction set architecture** (**ISA**) is an abstract model of a computer. It is also referred to as **architecture** or **computer architecture**. A realization of an ISA is called an *implementation*. An ISA permits multiple implementations that may vary in performance, physical size, and monetary cost (among other things); because the ISA serves as the interface between software and hardware. Software that has been written for an ISA can run on different implementations of the same ISA. This has enabled binary compatibility between different generations of computers to be easily achieved, and the development of computer families. Both of these developments have helped to lower the cost of computers and to increase their applicability. For these reasons, the ISA is one of the most important abstractions in computing today.

An ISA defines everything a machine language programming needs to know in order to program a computer. What an ISA defines differs between ISAs; in general, ISAs define the supported data types, what state there is (such as the memory addresses and registers) and their semantics (such as the memory consistency and addressing modes), the instruction 0set (the set of machine instructions that comprises a computer's machine language), and the i/o model.

* **CLASSIFICATION OF ISAs**

An ISA may be classified in a number of different ways. A common classification is by architectural *complexity*. A complex instruction set computer (CISC) has many specialized instructions, some of which may only be rarely used in practical programs. A reduced instruction set computer (RISC) simplifies the processor by efficiently implementing only the instructions that are frequently used in programs, while the less common operations are implemented as subroutines, having their resulting additional processor execution time offset by infrequent use.

Other types include very long instruction word (VLIW) architectures, and the closely related *long instruction word* (LIW) and explicitly parallel instruction computing (EPIC) architectures. These architectures seek to exploit instruction level parallelism with less hardware than RISC and CISC by making the compiler responsible for instruction issue and scheduling.

Architectures with even less complexity have been studied, such as the minimal instruction set computer (MISC) and one instruction set computer (OISC). These are theoretically important types, but have not been commercialized.

* **INSTRUCTIONS**

Machine-Language  is built up from discrete *statements* or *instructions*. On the processing architecture, a given instruction may specify:

* Particular registers (for arithmetic, addressing, or control functions)
* particular memory locations (or offsets to them)
* particular addressing modes (used to interpret the operands)

More complex operations are built up by combining these simple instructions, which are executed sequentially, or as otherwise directed by control flow instructions.

### INSTRUCTION TYPE

Examples of operations common to many instruction sets include:

* **DATA HANDLING AND MEMORY OPERATIONS**
* Set a register to a fixed constant value.
* Copy data from a memory location to a register, or vice versa (a machine instruction is often called move; however, the term is misleading). Used to store the contents of a register, the result of a computation or to retrieve stored data to perform a computation on it later. Often called load and store operations.
* Read and write data from hardware devices.

#### [ARITHMETIC AND LOGIC](https://en.wikipedia.org/wiki/Arithmetic_logic_unit) OPERATIONS

* Add, subtract, multiply, or divide the values of two registers, placing the result in a register, possibly setting one or more condition code in a [status register](https://en.wikipedia.org/wiki/Status_register).
* Increment, decrement in some ISAs, saving operand fetch in trivial cases.
* Perform [bitwise operations](https://en.wikipedia.org/wiki/Bitwise_operation), e.g., taking the [conjunction](https://en.wikipedia.org/wiki/Logical_conjunction) and [disjunction](https://en.wikipedia.org/wiki/Logical_disjunction) of corresponding bits in a pair of registers, taking the [negation](https://en.wikipedia.org/wiki/Logical_negation) of each bit in a register.
* Compare two values in registers (for example, to see if one is less, or if they are equal).
* Floating-point instructions for arithmetic on floating-point numbers.

#### [CONTROL FLOW](https://en.wikipedia.org/wiki/Control_flow) OPERATIONS

* [Branch](https://en.wikipedia.org/wiki/Branch_(computer_science)) to another location in the program and execute instructions there.
* [Conditionally branch](https://en.wikipedia.org/wiki/Branch_predication) to another location if a certain condition holds.
* [Indirectly branch](https://en.wikipedia.org/wiki/Indirect_branch) to another location.
* [Call](https://en.wikipedia.org/wiki/Subroutine) another block of code, while saving the location of the next instruction as a point to return to.

#### [COPROCESSOR](https://en.wikipedia.org/wiki/Coprocessor) INSTRUCTIONS

* Load/store data to and from a coprocessor or exchanging with CPU registers.
* Perform coprocessor operations.

### COMPLEX INSTRUCTIONS

Processors may include "complex" instructions in their instruction set. A single "complex" instruction does something that may take many instructions on other computers such instructions are [typified](https://en.wikipedia.org/wiki/Typified) by instructions that take multiple steps, control multiple functional units, or otherwise appear on a larger scale than the bulk of simple instructions implemented by the given processor. Some examples of "complex" instructions include:

* Transferring multiple registers to or from memory (especially the [stack](https://en.wikipedia.org/wiki/Call_stack)) at once
* Moving large blocks of memory (e.g. [string copy](https://en.wikipedia.org/wiki/String_copy) or [DMA transfer](https://en.wikipedia.org/wiki/DMA_transfer))
* Complicated integer and floating-point arithmetic (e.g. [square root](https://en.wikipedia.org/wiki/Square_root), or [transcendental functions](https://en.wikipedia.org/wiki/Transcendental_function) such as [logarithm](https://en.wikipedia.org/wiki/Logarithm), [sine](https://en.wikipedia.org/wiki/Sine), [cosine](https://en.wikipedia.org/wiki/Cosine), etc.)
* [SIMD](https://en.wikipedia.org/wiki/SIMD) instructions, a single instruction performing an operation on many homogeneous values in parallel, possibly in dedicated [SIMD registers](https://en.wikipedia.org/wiki/SIMD_register)
* Performing an atomic [test-and-set](https://en.wikipedia.org/wiki/Test-and-set) instruction or other [read-modify-write](https://en.wikipedia.org/wiki/Read-modify-write) [atomic instruction](https://en.wikipedia.org/wiki/Atomic_instruction)
* Instructions that perform [ALU](https://en.wikipedia.org/wiki/Arithmetic_logic_unit) operations with an operand from memory rather than a register.

Complex instructions are more common in CISC instruction sets than in RISC instruction sets, but RISC instruction sets may include them as well. RISC instruction sets generally do not include ALU operations with memory operands, or instructions to move large blocks of memory, but most RISC instruction sets include [SIMD](https://en.wikipedia.org/wiki/SIMD) or [vector](https://en.wikipedia.org/wiki/Vector_processing) instructions that perform the same arithmetic operation on multiple pieces of data at the same time. SIMD instructions have the ability of manipulating large vectors and matrices in minimal time. SIMD instructions allow easy [parallelization](https://en.wikipedia.org/wiki/Parallelization) of algorithms commonly involved in sound, image, and video processing. Various SIMD implementations have been brought to market under trade names such as [MMX](https://en.wikipedia.org/wiki/MMX_(instruction_set)), [3DNow!](https://en.wikipedia.org/wiki/3DNow!).

* **INSTRUCTION ENCODING**

On traditional architectures, an instruction includes an [opcode](https://en.wikipedia.org/wiki/Opcode) that specifies the operation to perform, such as *add contents of memory to register*—and zero or more [operand](https://en.wikipedia.org/wiki/Operand) specifiers, which may specify [registers](https://en.wikipedia.org/wiki/Processor_register), memory locations, or literal data. The operand specifiers may have [addressing modes](https://en.wikipedia.org/wiki/Addressing_mode) determining their meaning or may be in fixed fields. In [very long instruction word](https://en.wikipedia.org/wiki/Very_long_instruction_word) (VLIW) architectures, which include many [microcode](https://en.wikipedia.org/wiki/Microcode) architectures, multiple simultaneous opcodes and operands are specified in a single instruction.

Some exotic instruction sets do not have an opcode field, such as [transport triggered architectures](https://en.wikipedia.org/wiki/Transport_triggered_architecture) (TTA), only operand(s).

The [Forth virtual machine](https://en.wikipedia.org/wiki/Forth_virtual_machine) and other "[0-operand](https://en.wikipedia.org/wiki/0-operand_instruction_set)" instruction sets lack any operand specifier fields, such as some [stack machines](https://en.wikipedia.org/wiki/Stack_machine) including NOSC.

Conditional instructions often have a predicate field—a few bits that encode the specific condition to cause the operation to be performed rather than not performed. For example, a conditional branch instruction will be executed, and the branch taken, if the condition is true, so that execution proceeds to a different part of the program, and not executed, and the branch not taken, if the condition is false, so that execution continues sequentially. Some instruction sets also have conditional moves, so that the move will be executed, and the data stored in the target location, if the condition is true, and not executed, and the target location not modified, if the condition is false. Similarly, IBM [z/Architecture](https://en.wikipedia.org/wiki/Z/Architecture) has a conditional store instruction. A few instruction sets include a predicate field in every instruction; this is called [branch predication](https://en.wikipedia.org/wiki/Branch_predication).

#### NUMBER OF OPERANDS

Instruction sets may be categorized by the maximum number of operands explicitly specified in instructions.

(In the examples that follow, A, B, and C are (direct or calculated) addresses referring to memory cells, while reg1 and so on refer to machine registers.)

C = A+B

* 0-operand (zero-address machines), so called [stack machines](https://en.wikipedia.org/wiki/Stack_machine): All arithmetic operations take place using the top one or two positions on the stack: push a, push b, add, pop c.
* C = A+B needs four instructions. For stack machines, the terms "0-operand" and "zero-address" apply to arithmetic instructions, but not to all instructions, as 1-operand push and pop instructions are used to access memory.

1. 1-operand (one-address machines), so called [accumulator machines](https://en.wikipedia.org/wiki/Accumulator_machine), include early computers and many small [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller): most instructions specify a single right operand (that is, constant, a register, or a memory location), with the implicit [accumulator](https://en.wikipedia.org/wiki/Accumulator_(computing)) as the left operand (and the destination if there is one): load a, add b, store c.

* C = A+B needs three instructions.

1. 2-operand — many CISC and RISC machines fall under this category:

* CISC — move A to C; then add B to C.
* C = A+B needs two instructions. This effectively 'stores' the result without an explicit store instruction.
* CISC — Often machines are [limited to one memory operand](https://web.archive.org/web/20131105155703/http:/cs.smith.edu/~thiebaut/ArtOfAssembly/CH04/CH04-3.html#HEADING3-79) per instruction: load a,reg1; add b,reg1; store reg1,c; This requires a load/store pair for any memory movement regardless of whether the add result is an augmentation stored to a different place, as in C = A+B, or the same memory location: A = A+B.
* C = A+B needs three instructions.
* RISC — Requiring explicit memory loads, the instructions would be: load a,reg1; load b,reg2; add reg1,reg2; store reg2,c.
  + - C = A+B needs four instructions.

1. 3-operand, allowing better reuse of data:[[4]](https://en.wikipedia.org/wiki/Instruction_set_architecture#cite_note-Cocke-4)

* CISC — It becomes either a single instruction: add a,b,c
  + - C = A+B needs one instruction.
* CISC — Or, on machines limited to two memory operands per instruction, move a,reg1; add reg1,b,c;
* C = A+B needs two instructions.
* RISC — arithmetic instructions use registers only, so explicit 2-operand load/store instructions are needed: load a,reg1; load b,reg2; add reg1+reg2->reg3; store reg3,c;
* C = A+B needs four instructions.
* Unlike 2-operand or 1-operand, this leaves all three values a, b, and c in registers available for further reuse.[[4]](https://en.wikipedia.org/wiki/Instruction_set_architecture#cite_note-Cocke-4)
* More operands—some CISC machines permit a variety of addressing modes that allow more than 3 operands (registers or memory accesses), such as the [VAX](https://en.wikipedia.org/wiki/VAX) "POLY" polynomial evaluation instruction.

Due to the large number of bits needed to encode the three registers of a 3-operand instruction, RISC architectures that have 16-bit instructions are invariably 2-operand designs, such as the Atmel AVR, [TI MSP430](https://en.wikipedia.org/wiki/TI_MSP430), and some versions of [ARM Thumb](https://en.wikipedia.org/wiki/ARM_Thumb). RISC architectures that have 32-bit instructions are usually 3-operand designs, such as the [ARM](https://en.wikipedia.org/wiki/ARM_architecture), [AVR32](https://en.wikipedia.org/wiki/AVR32), [MIPS](https://en.wikipedia.org/wiki/MIPS_architecture), [Power ISA](https://en.wikipedia.org/wiki/Power_ISA), and [SPARC](https://en.wikipedia.org/wiki/SPARC) architectures.

Each instruction specifies some number of operands (registers, memory locations, or immediate values) explicitly. Some instructions give one or both operands implicitly, such as by being stored on top of the [stack](https://en.wikipedia.org/wiki/Stack_(data_structure)) or in an implicit register. If some of the operands are given implicitly, fewer operands need be specified in the instruction. When a "destination operand" explicitly specifies the destination, an additional operand must be supplied. Consequently, the number of operands encoded in an instruction may differ from the mathematically necessary number of arguments for a logical or arithmetic operation. Operands are either encoded in the "Opcode" representation of the instruction, or else are given as values or addresses following the instruction.

* **INSTRUCTION SET IMPLEMENTATION**

Any given instruction set can be implemented in a variety of ways. All ways of implementing a particular instruction set provide the same [programming model](https://en.wikipedia.org/wiki/Programming_model), and all implementations of that instruction set are able to run the same executable. The various ways of implementing an instruction set give different tradeoffs between cost, performance, power consumption, size, etc.

When designing the [microarchitecture](https://en.wikipedia.org/wiki/Microarchitecture) of a processor, engineers use blocks of "hard-wired" electronic circuitry (often designed separately) such as adders, multiplexers, counter, registers, ALUs, etc. Some kind of [register transfer language](https://en.wikipedia.org/wiki/Register_transfer_language) is then often used to describe the decoding and sequencing of each instruction of an ISA using this physical microarchitecture. There are two basic ways to build a [control unit](https://en.wikipedia.org/wiki/Control_unit) to implement this description (although many designs use middle ways or compromises):

Some computer designs "hardwire" the complete instruction set decoding and sequencing (just like the rest of the microarchitecture).

Other designs employ [microcode](https://en.wikipedia.org/wiki/Microcode) routines or tables (or both) to do this—typically as on-chip [ROMs](https://en.wikipedia.org/wiki/Read-only_memory) or [PLAs](https://en.wikipedia.org/wiki/Programmable_logic_array) or both (although separate RAMs and [ROMs](https://en.wikipedia.org/wiki/Read-only_memory#Historical_examples) have been used historically). The [Western Digital](https://en.wikipedia.org/wiki/Western_Digital) [MCP-1600](https://en.wikipedia.org/wiki/MCP-1600) is an older example, using a dedicated, separate ROM for microcode.

Some designs use a combination of hardwired design and microcode for the control unit.

Some CPU designs use a [writable control store](https://en.wikipedia.org/wiki/Writable_control_store)—they compile the instruction set to a writable [RAM](https://en.wikipedia.org/wiki/RAM) or [flash](https://en.wikipedia.org/wiki/Flash_memory) inside the CPU (such as the [Recursive](https://en.wikipedia.org/wiki/Rekursiv) processor and the [Imsys](https://en.wikipedia.org/w/index.php?title=Imsys&action=edit&redlink=1" \o "Imsys (page does not exist)) Chip),[[10]](https://en.wikipedia.org/wiki/Instruction_set_architecture#cite_note-10) or an FPGA ([reconfigurable computing](https://en.wikipedia.org/wiki/Reconfigurable_computing)).

An ISA can also be [emulated](https://en.wikipedia.org/wiki/Emulator) in software by an [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)). Naturally, due to the interpretation overhead, this is slower than directly running programs on the emulated hardware, unless the hardware 1running the emulator is an order of magnitude faster. Today, it is common practice for vendors of new ISAs or microarchitectures to ma1ke software emulators available to software developers before the hardware implementation is ready.

Often the details of the implementation have a strong influence on the particular instructions selected for the instruction set. For example, many implementations of the [instruction pipeline](https://en.wikipedia.org/wiki/Instruction_pipeline) only allow a single memory load or memory store per instruction, leading to a  [load-store architecture](https://en.wikipedia.org/wiki/Load-store_architecture) (RISC). For another example, some early ways of implementing the [instruction pipeline](https://en.wikipedia.org/wiki/Instruction_pipeline) led to a [delay slot](https://en.wikipedia.org/wiki/Delay_slot).

The demands of high-speed digital signal processing have pushed in the opposite direction—forcing instructions to be implemented in a particular way. For example, to perform digital filters fast enough, the MAC instruction in a typical [digital signal processor](https://en.wikipedia.org/wiki/Digital_signal_processor) (DSP) must use a kind of [Harvard architecture](https://en.wikipedia.org/wiki/Harvard_architecture) that can fetch an instruction and two data words simultaneously, and it requires a single-cycle [multiply–accumulate](https://en.wikipedia.org/wiki/Multiply%E2%80%93accumulate) [multiplier](https://en.wikipedia.org/wiki/Binary_multiplier).

**MEMORY ORGANIZATION**

**Memory** is the faculty of the [brain](https://en.wikipedia.org/wiki/Brain) by which [data](https://en.wikipedia.org/wiki/Data) or [information](https://en.wikipedia.org/wiki/Information) is encoded, stored, and retrieved when needed. It is the retention of information over time for the purpose of influencing future action. If [past events](https://en.wikipedia.org/wiki/Foresight_(psychology)) could not be remembered, it would be impossible for language, relationships, or [personal identity](https://en.wikipedia.org/wiki/Personal_identity) to develop. Memory loss is usually described as [forgetfulness](https://en.wikipedia.org/wiki/Forgetting) or [amnesia](https://en.wikipedia.org/wiki/Amnesia).

Memory is often understood as an [informational processing](https://en.wikipedia.org/wiki/Information_processing) system with explicit and implicit functioning that is made up of a [sensory processor](https://en.wikipedia.org/wiki/Sensory_processor), [short-term](https://en.wikipedia.org/wiki/Short-term_memory) (or [working](https://en.wikipedia.org/wiki/Working_memory)) memory, and [long-term memory](https://en.wikipedia.org/wiki/Long-term_memory). This can be related to the [neuron](https://en.wikipedia.org/wiki/Neuron). The sensory processor allows information from the outside world to be sensed in the form of chemical and physical stimuli and attended to various levels of focus and intent. Working memory serves as an encoding and retrieval processor. Information in the form of stimuli is encoded in accordance with explicit or implicit functions by the working memory processor. The working memory also retrieves information from previously stored material. Finally, the function of long-term memory is to store data through various categorical models or systems.

[Declarative, or explicit, memory](https://en.wikipedia.org/wiki/Explicit_memory) is the conscious storage and recollection of data. Under declarative memory resides [semantic](https://en.wikipedia.org/wiki/Semantic_memory) and [episodic memory](https://en.wikipedia.org/wiki/Episodic_memory). Semantic memory refers to memory that is encoded with specific meaning while episodic memory refers to information that is encoded along a spatial and temporal plane. Declarative memory is usually the primary process thought of when referencing memory. [Non-declarative, or implicit, memory](https://en.wikipedia.org/wiki/Implicit_memory) is the unconscious storage and recollection of information. An example of a non-declarative process would be the unconscious learning or retrieval of information by way of [procedural memory](https://en.wikipedia.org/wiki/Procedural_memory), or a priming phenomenon. [Priming](https://en.wikipedia.org/wiki/Priming_(psychology)) is the process of [subliminally](https://en.wikipedia.org/wiki/Subliminal_stimuli) arousing specific responses from memory and shows that not all memory is consciously activated. whereas procedural memory is the slow and gradual learning of skills that often occurs without conscious attention to learning.

Memory is not a perfect processor, and is affected by many factors. The ways by which information is encoded, stored, and retrieved can all be corrupted? The amount of attention given new stimuli can diminish the amount of information that becomes encoded for storage. Also, the storage process can become corrupted by physical damage to areas of the brain that are associated with memory storage, such as the hippocampus. Finally, the retrieval of information from long-term memory can be disrupted because of decay within long-term memory. Normal functioning, decay over time, and brain damage all affect the accuracy and capacity of the memory

**CONTROL UNIT**

Definition – What does Control Unit (CU*)* mean?

A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions from micro programs and directs other units and models by providing control and timing signals. A CU component is considered the processor brain because it issues orders to just about everything and ensures correct instruction execution.

A CU takes its input from the instruction and status registers. Its rules of operation, or micro program, are encoded in a programmable logic array (PLA), random logic or read-only memory.  
  
CU functions are as follows:

* Controls sequential instruction execution
* Interprets instructions
* Guides data flow through different mobile areas
* Regulates and controls processor timing
* Sends and receives control signals from other mobile devices
* Handles multiple tasks, such as fetching, decoding, execution handling and storing results

Control Unit is designed in two ways:

* Hardwired control: Design is based on a fixed architecture. The CU is made up of flip-flops, logic gates, digital circuits and encoder and decoder circuits that are wired in a specific and fixed way. When instruction set changes are required, wiring and circuit changes must be made. This is preferred in a reduced instruction set computing (RISC) architecture, which only has a small number of instructions.
* Micro program control: Micro programs are stored in a special control memory and are based on flowcharts. They are replaceable and ideal because of their simplicity.

**I/O MECHANISMS**

In mobile computing, **input/output** or **I/O** is the communication between an [information processing system](https://en.wikipedia.org/wiki/Information_processing_system), such as a mobile and the outside world, possibly a human or another [information processing system](https://en.wikipedia.org/wiki/Information_processor). [Inputs](https://en.wikipedia.org/wiki/Information) are the signals or data received by the system and outputs are the signals or [data](https://en.wikipedia.org/wiki/Data_(computing)) sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an [input or output operation](https://en.wikipedia.org/wiki/I/O_scheduling).

**I/O devices** are the pieces of [hardware](https://en.wikipedia.org/wiki/Hardware_(computing)) used by a human (or other system) to communicate with a mobile. For instance, a keypad or keyboard is an [input device](https://en.wikipedia.org/wiki/Input_device) for a mobile, while display is output devices. Devices for communication between mobiles, such as internet connection typically perform both input and output operations.

The designation of a device as either input or output depends on perspective. keypad or keyboards take physical movements that the human user outputs and convert them into input signals that a mobile can understand; the output from these devices is the mobile’s input. Similarly, monitors or display take signals that a mobile outputs as input, and they convert these signals into a representation that human users can understand. From the human [user](https://en.wikipedia.org/wiki/User_(computing))'s perspective, the process of reading or seeing these representations is receiving output; this type of interaction between mobiles and humans is studied in the field of [human-mobiles interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction).

In computer and mobile architecture, the combination of the [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) and [main memory](https://en.wikipedia.org/wiki/Main_memory), to which the CPU can read or write directly using individual [instructions](https://en.wikipedia.org/wiki/Instruction_(computer_science)), is considered the brain of a computer. Any transfer of information to or from the CPU/memory combo, for example by reading data from a [disk drive](https://en.wikipedia.org/wiki/Disk_drive), is considered I/O.[[1]](https://en.wikipedia.org/wiki/Input/output#cite_note-teco-1) The CPU and its supporting circuitry may provide [memory-mapped I/O](https://en.wikipedia.org/wiki/Memory-mapped_I/O) that is used in low-level [computer programming](https://en.wikipedia.org/wiki/Computer_programming), such as in the implementation of [device drivers](https://en.wikipedia.org/wiki/Device_driver), or may provide access to [I/O channels](https://en.wikipedia.org/wiki/Channel_I/O). An [I/O algorithm](https://en.wikipedia.org/wiki/External_memory_algorithm) is one designed to exploit locality and perform efficiently when exchanging data with a secondary storage device, such as a disk drive.

* **INTERFACE**

An I/O interface is required whenever the I/O device is driven by a processor. Typically a CPU communicates with devices via a [bus](https://en.wikipedia.org/wiki/Bus_(computing)). The interface must have necessary logic to interpret the device address generated by the processor. [Handshaking](https://en.wikipedia.org/wiki/Handshaking) should be implemented by the interface using appropriate commands (like BUSY, READY, and WAIT), and the processor can communicate with an I/O device through the interface. If different data formats are being exchanged, the interface must be able to convert serial data to parallel form and vice versa. Because it would be a waste for a processor to be idle while it waits for data from an input device there must be provision for generating [interrupts](https://en.wikipedia.org/wiki/Interrupt) and the corresponding type numbers for further processing by the processor if required.

A mobile that uses [memory-mapped I/O](https://en.wikipedia.org/wiki/Memory-mapped_I/O) accesses hardware by reading and writing to specific memory locations, using the same assembly language instructions that mobile would normally use to access memory. An alternative method is via instruction-based I/O which requires that a CPU have specialized instructions for I/O. Both input and output devices have a [data processing](https://en.wikipedia.org/wiki/Data_processing) rate that can vary greatly. With some devices able to exchange data at very high speeds [direct access](https://en.wikipedia.org/wiki/Direct_memory_access) to memory (DMA) without the continuous aid of a CPU is required

**CONTROL UNIT**

Definition – What does Control Unit (CU*)* mean?

A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions from micro programs and directs other units and models by providing control and timing signals. A CU component is considered the processor brain because it issues orders to just about everything and ensures correct instruction execution.

A CU takes its input from the instruction and status registers. Its rules of operation, or micro program, are encoded in a programmable logic array (PLA), random logic or read-only memory.  
  
CU functions are as follows:

* Controls sequential instruction execution
* Interprets instructions
* Guides data flow through different mobile areas
* Regulates and controls processor timing
* Sends and receives control signals from other mobile devices
* Handles multiple tasks, such as fetching, decoding, execution handling and storing results

Control Unit is designed in two ways:

* Hardwired control: Design is based on a fixed architecture. The CU is made up of flip-flops, logic gates, digital circuits and encoder and decoder circuits that are wired in a specific and fixed way. When instruction set changes are required, wiring and circuit changes must be made. This is preferred in a reduced instruction set computing (RISC) architecture, which only has a small number of instructions.
* Micro program control: Micro programs are stored in a special control memory and are based on flowcharts. They are replaceable and ideal because of their simplicity.

**Legal Information**

Native name : 联想集团有限公司  
Type : Public

Traded as : SEHK: [992](tel:992) OTC Pink: LNVGY

Industry : Computer hardware Electronics

Founded : 1 November [1984](tel:1984); 34 years ago (as Legend)Beijing

Founder: Liu Chuanzhi

Headquarters Haidian District, Beijing

Area served: Worldwide

Key people:  
Yang Yuanqing(Chairman & CEO)  
Gianfranco Lanci (COO)  
Yong Rui (CTO)  
Wong Wai Ming (CFO)

Products : Smartphones, desktops, servers, notebooks, tablet computers, netbooks, supercomputers, peripherals, printers, televisions, scanners, storage devices

Revenue:  US$[51.038](tel:51.038) billion ([2019](tel:2019))  
  
Operating income: US$[1.178](tel:1.178) million ([2019](tel:2019))[1]  
  
Net income : US$[597](tel:597) million ([2019](tel:2019))

Total asset :  US$[28.49](tel:28.49) billion ([2018](tel:2018))

Total equity : US$[51.038](tel:51.038) billion ([2019](tel:2019))

Owner : Legend Holdings([30.6](tel:30.6)%)  
  
Number of employees : 54,[000](tel:000) ([2018](tel:2018))

Subsidiaries : Motorola Mobility ZUK Mobile medion

Website : www.lenovo.com