Unit-02/Lecture-01

Control Unit Organization

Control unit

The control unit is a component of a computer's central processing unit (CPU) that directs operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices how to respond to a program's instructions.

It directs the operation of the other units by providing timing and control signals.[citation needed] All computer resources are managed by the CU (Control Unit).[citation needed] It directs the flow of data between the Central Processing Unit (CPU) and the other devices. John von Neumann included the control unit as part of the von Neumann architecture. In modern computer designs, the control unit is typically an internal part of the CPU with its overall role and operation unchanged.

Functions of the Control Unit

The Control Unit is the circuitry that controls the flow of data through the processor, and coordinates the activities of the other units within it.[citation needed] In a way, it is the "brain within the brain", as it controls what happens inside the processor, which in turn controls the rest of the computer.[citation needed] The examples of devices that require a Control Unit are CPUs and graphics processing units (GPUs).[citation needed] The Control Unit receives external instructions or commands which it converts into a sequence of control signals that the Control Unit applies to the data path to implement a sequence of register-transfer level operations Control Unit.

The Control Unit (CU) is generally a sizable collection of complex digital circuitry interconnecting and controlling the many execution units contained within a CPU.[citation needed] The CU is normally the first CPU unit to accept from an externally stored computer program, a single instruction, based on the CPU’s instruction set, then decode this individual instruction into several sequential steps (fetching addresses/data from registers/memory, managing execution [i.e. data sent to the ALU or I/O], and storing the resulting data back into registers/memory) that controls and coordinates the CPU’s interworks.[citation needed] These detailed steps from the CU dictate which of the numerous CPU's interconnecting hardware control signals to enable/disable or which CPU units are selected/de-selected and the unit's proper order of execution as required by the instruction's operation.[citation needed] Additionally, the CU's orderly hardware coordination properly sequences these control signals then configures the many hardware units comprising the CPU, directing how data should also be moved, changed, and stored outside the CPU (i.e. memory) according to the instruction's objective.[citation needed] Depending on the type of instruction entering the CU, the order and number of sequential steps produced by the CU could vary the selection and configuration of which parts of the CPU's hardware are utilized to achieve the instruction's objective (mainly moving, storing, and modifying data within the CPU).[citation needed] This one feature, that efficiently uses just software instructions to control/select/configure a computer's CPU hardware (via the CU) and eventually manipulates a program's data, is a significant reason most modern computers are flexible and universal when running various programs. As compared to some 1930s or 1940s computers without a proper CU, they often required rewiring their hardware when changing programs.[citation needed] This CU instruction decode process is then repeated when the Program Counter is incremented to the next stored program address and the new instruction enters the CU from that address, and so on till the programs end.

Other more advanced forms of Control Units manage the translation of instructions (but not the data containing portion) into several micro-instructions and the CU manages the scheduling of the micro-instructions between the selected execution units to which the data is then channeled and changed according to the execution unit's function (i.e., ALU contains several functions).[citation needed] On some processors, the Control Unit may be further broken down into additional units, such as an instruction unit or scheduling unit to handle scheduling, or a retirement unit to deal with results coming from the instruction pipeline.[citation needed] Again, the Control Unit orchestrates the main functions of the CPU: carrying out stored instructions in the software program then directing the flow of data throughout the computer based upon these instructions (roughly likened to how traffic lights will systematically control the flow of cars [containing data] to different locations within the traffic grid [CPU] until it parks at the desired parking spot [memory address/register].[citation needed] The car occupants [data] then go into the building [execution unit] and comes back changed in some way then get back into the car and returns to another location via the controlled traffic grid).

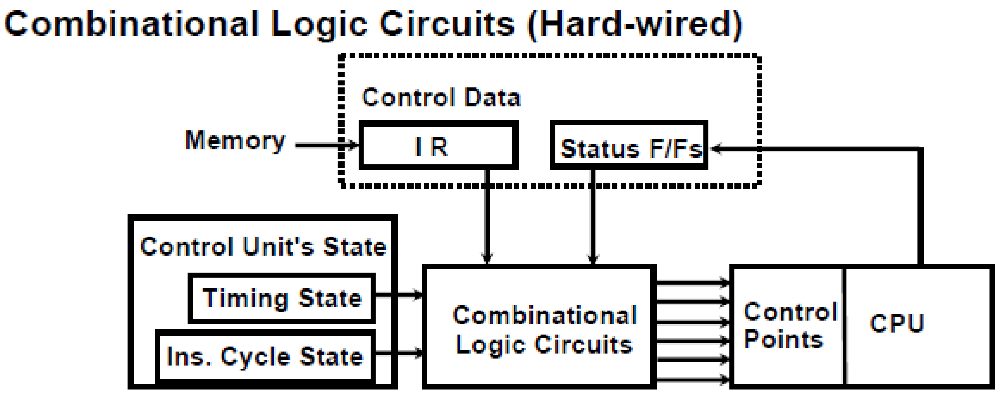


Fig 2.1 Combinational Logic Circuits

Hardwired control unit

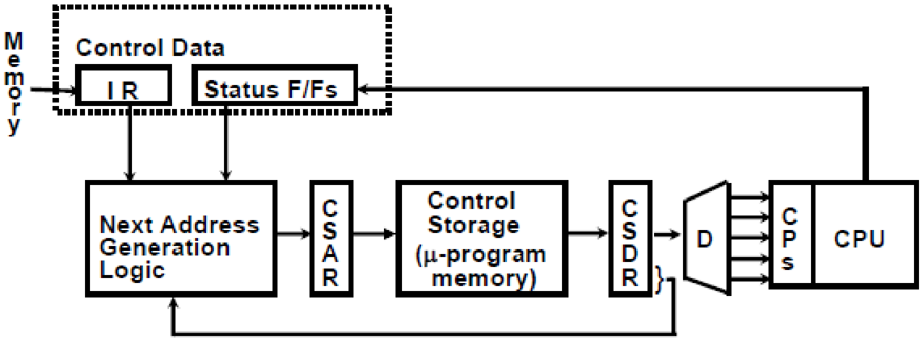
Hardwired control units are implemented through use of sequential logic units, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses. Hardwired control units are generally faster than micro programmed designs. Their design uses a fixed architecture—it requires changes in the wiring if the instruction set is modified or changed.[citation needed] This architecture is preferred in reduced instruction set computers (RISC) as they use a simpler instruction set.

A controller that uses this approach can operate at high speed; however, it has little flexibility, and the complexity of the instruction set it can implement is limited.

The hardwired approach has become less popular as computers have evolved. Previously, control units for CPUs used ad-hoc logic, and they were difficult to design.

Micro program control unit:

The idea of microprogramming was introduced by Maurice Wilkes in 1951 as an intermediate level to execute computer program instructions. Micro programs were organized as a sequence of microinstructions and stored in special control memory. The algorithm for the micro program control unit is usually specified by flowchart description.[4] The main advantage of the micro program control unit is the simplicity of its structure. Outputs of the controller are organized in microinstructions and they can be easily replaced.



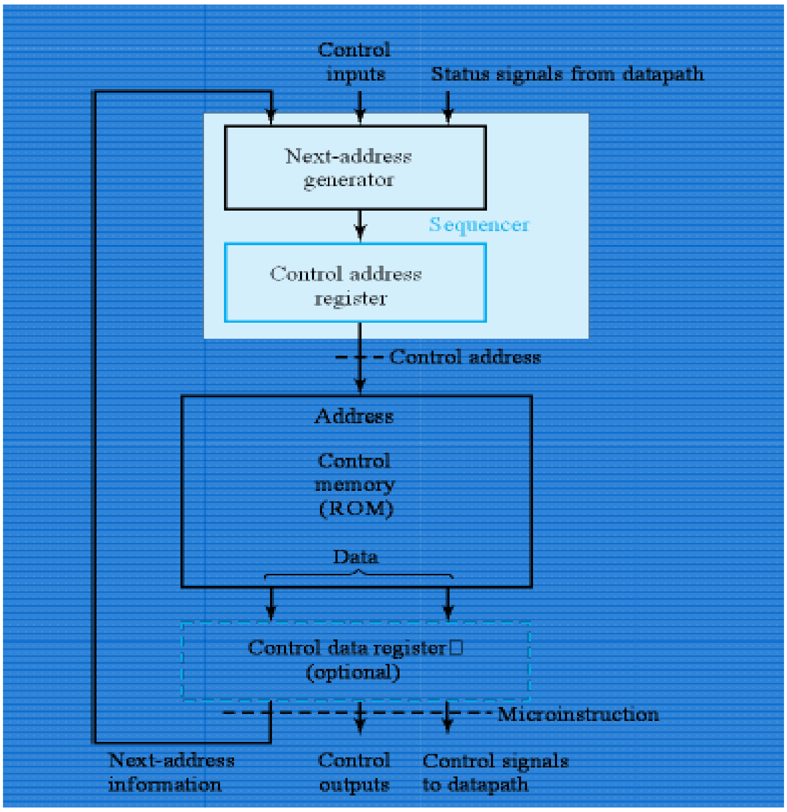


Fig 2.2S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Discuss in brief microprogram control unit and hardwired control unit?	June 2012	7
Q.2	What is need of control unit in a computer?	Dec 2010	4

Unit-02/Lecture-02

Control Memory

- The control unit in a digital computer initiates sequences of micro operations.
- The complexity of the digital system is derived form the number of sequences that are performed
- When the control signals are generated by hardware, it is hardwired
- In a bus-oriented system, the control signals that specify micro operations are groups of bits that select the paths in multiplexers, decoders, and ALUs.
- The control unit initiates a series of sequential steps of micro operations
- The control variables can be represented by a string of 1's and 0's called a control word
- A micro programmed control unit is a control unit whose binary control variables are stored in memory .
- The control unit initiates a series of sequential steps of micro operations
- The control variables can be represented by a string of 1's and 0's called a
- control word
- A micro programmed control unit is a control unit whose binary control variables are stored in memory
- A sequence of microinstructions constitutes a micro program
- The control memory can be a read-only memory
- Dynamic microprogramming permits a micro program to be loaded and uses a writable control memory
- A computer with a micro programmed control unit will have two separate
- memories: a main memory and a control memory
- The micro program consists of microinstructions that specify various internal control signals for execution of register micro operations
- These microinstructions generate the micro operations to:

- fetch the instruction from main memory
- evaluate the effective address

- execute the operation
- Return control to the fetch phase for the next instruction.
- The control memory address register specifies the address of the microinstruction
- The control data register holds the microinstruction read from memory
- The microinstruction contains a control word that specifies one or more
- Micro operations for the data processor

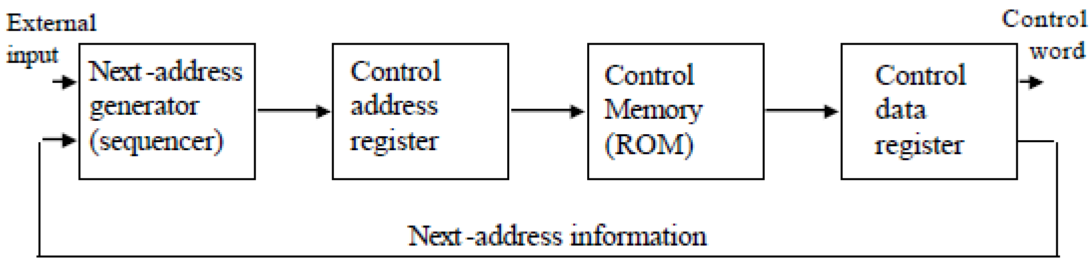


Fig 2.3 Control unit

Address Sequencing

Microinstructions are usually stored in groups where each group specifies a routine, where each routine specifies how to carry out an instruction. Each routine must be able to branch to the next routine in the sequence. An initial address is loaded into the CAR when power is turned on; this is usually the address of the first microinstruction in the instruction fetch routine. Next, the control unit must determine the effective address of the instruction.

When instruction execution is finished, control must be return to the fetch routine. This is done using an unconditional branch. Addressing sequencing capabilities of control memory include Incrementing the CAR Unconditional and conditional branching (depending on status bit).Mapping instruction bits into control memory addresses Handling subroutine calls and returns.

Mapping

The next step is to generate the micro operations that executed the instruction. This involves taking the instructions op code and transforming it into an address for the instructions micro program in control memory. This process is called mapping.

While microinstruction sequences are usually determined by incrementing the CAR, this is not always the case. If the processor's control unit can support subroutines in a micro program, it will need an external register for storing return addresses

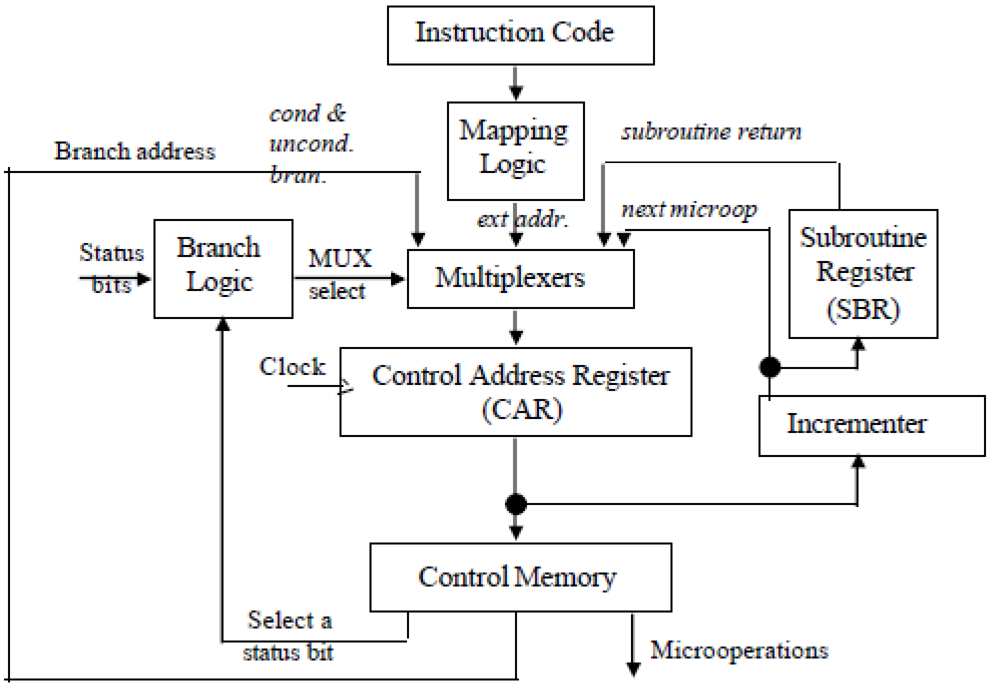


Fig 2.4 Selection of Address For control memory

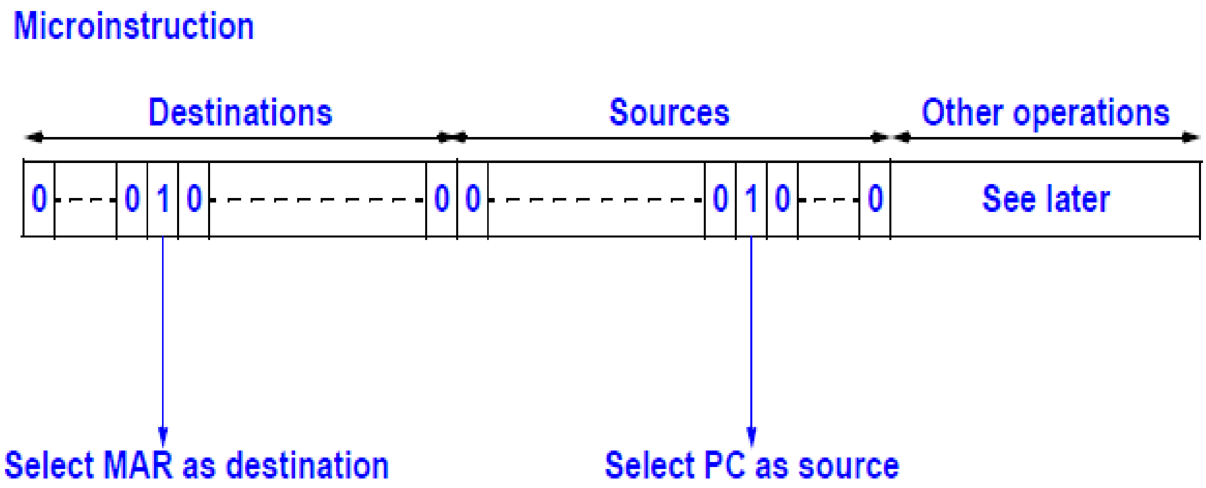
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Explain the concept of address sequencing .Also explain mapping of an instruction.	June 2012	7

Microinstruction Formats:

Two basic formats

1. **Horizontal Microinstruction Format**

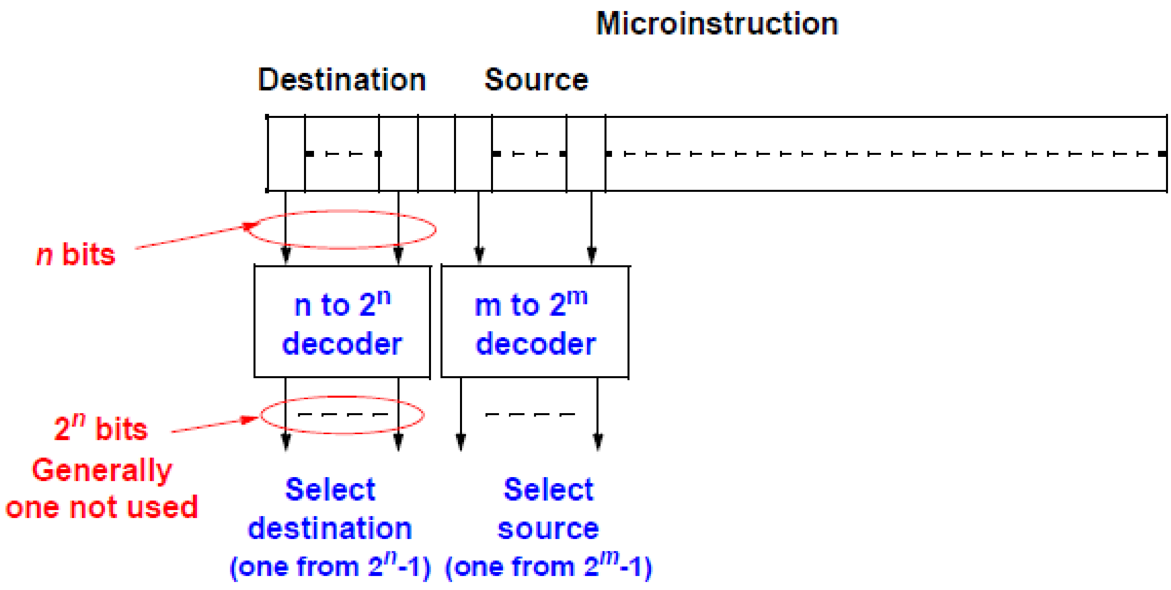
One bit for each possible signal that might need to be generated by any microinstruction -leads to the fastest execution:

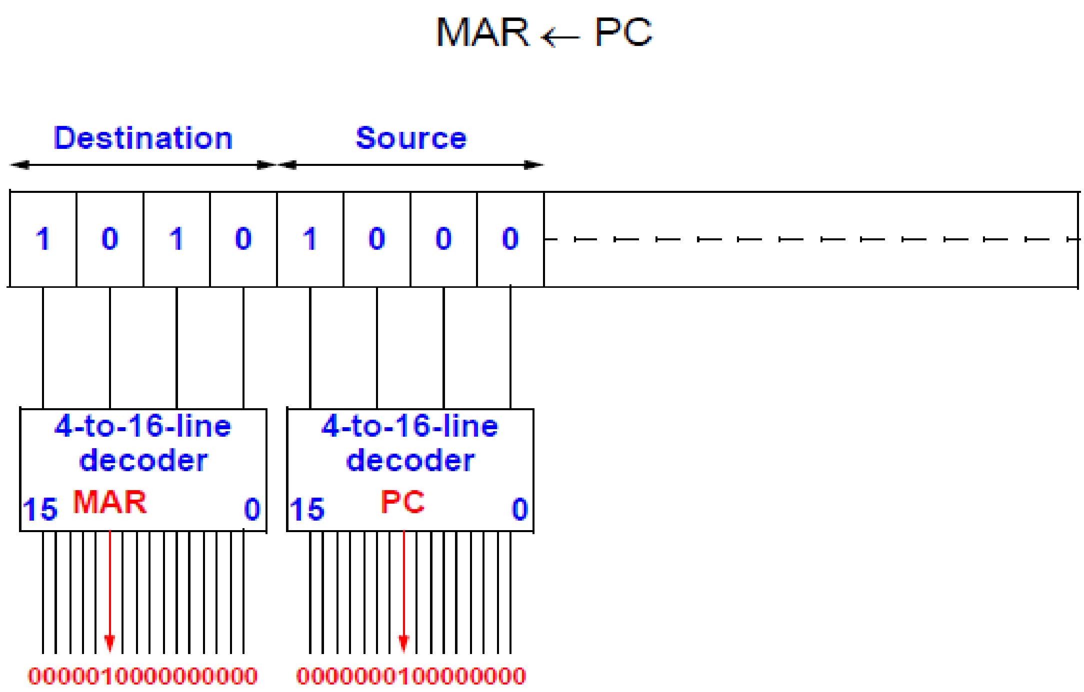


Requires d bits if there are d possible destinations plus s bits if there are s possible sources.

2. **Vertical Microinstruction Format**

Mutually exclusive operations grouped together and encoded in binary. Reduces number of bits in microinstruction. Each vertically encoded field needs a decoder





Example

Suppose there were up to 15 possible sources and destinations (PC, MAR, MDR, IR ....). Four bits needed to specify which one:

Encoding	Destination	Source	
0000	None	None	One pattern for no signal.
0001	Rd	Rs1	
0010	.	Rs2	
0011	IR	.	
0100	.	IR_{15-0}	Various parts of IR
0101	.	IR_{25-0}	
0110	.	0 (zero)	Used in branch
0111	R31	4 (constant)	Used to increment PC
1000	PC	PC	
1001	MDR	MDR	
1010	MAR	MAR	
.	.	.	

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Define the following: (i) Micro –operation (ii) Microcode (iii) Microinstruction (iv) MicroProgram	June 2012	7

Unit-02/Lecture-04

Micro Program sequencer

In computer architecture and engineering, a sequencer or micro sequencer generates the addresses used to step through the micro program of a control store. It is used as a part of the control unit of a CPU or as a stand-alone generator for address ranges.

Usually the addresses are generated by some combination of a counter, a field from a microinstruction, and some subset of the instruction register. A counter is used for the typical case, that the next microinstruction is the one to execute. A field from the microinstruction is used for jumps, or other logic.

Since CPUs implement an instruction set, it's very useful to be able to decode the instruction's bits directly into the sequencer, to select a set of microinstructions to perform a CPU's instructions.

Most modern CPUs are considerably more complex than this description suggests. They tend to have multiple cooperating micro machines with specialized logic to detect and handle interference between the micro machines.

Or

A micro program sequencer for micro programmed control unit develops micro program consecutive addresses, branches to subroutines with address saving and possible return to micro program, as well as interrupting micro program forcings with address saving of the interrupted micro programs. In order to allow the double saving of micro program and subroutine addresses in case of concurrent interruptions and branches, the sequencer is provided with two address generation loops each including a register. The two loops have a common portion to which they accede through a multiplexer (23). The first loop (23, 25, 22, 21, 30, 31) is further coupled to a saving register stack (20). While the first loop executes the saving of a micro program address and the latching of a branch address received from the second loop, the second loop (23, 25, 24, 39, 17, 18, 42, 19, 27, 29) executes a first updating and-, related latching of interrupting micro program address. During the following cycle, by command of the first microinstruction of the interrupting micro program, the second loop performs a first updating and related latch of the interrupting micro program address and the first loop saves into the register stack (20) the branch address and performs a second updating and related latching of the interrupting micro program address.

MultiProgramming

Microcode is a layer of hardware-level instructions that implement higher-level machine code instructions or internal state machine sequencing in many digital processing elements. Microcode is used in general central processing units, in more specialized processors such as microcontrollers, digital signal processors, channel controllers, disk controllers, network interface controllers, network processors, graphics processing units, and in other hardware.

Microcode typically resides in special high-speed memory and translates machine instructions, state machine data or other input into sequences of detailed circuit-level operations. It separates the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also facilitates the building of complex multi-step instructions, while reducing the complexity of computer circuits. Writing microcode is often called **microprogramming** and the microcode in a particular processor implementation is sometimes called a microprogram.

Engineers normally write the microcode during the design phase of a processor, storing it in a ROM (read-only memory) or PLA (programmable logic array)[1] structure, or in a combination of both. However, machines also exist that have some (or all) microcode stored in SRAM or flash memory. This is traditionally denoted a "writeable control store" in the context of computers. Complex digital processors may also employ more than one (possibly microcode-based) control unit in order to delegate sub-tasks that must be performed (more or less) asynchronously in parallel. A high-level programmer, or even an assembly programmer, does not normally see or change microcode. Unlike machine code, which often retains some compatibility among different processors in a family, microcode only runs on the exact electronic circuitry for which it is designed, as it constitutes an inherent part of the particular processor design itself

More extensive microcoding allows small and simple microarchitectures to emulate more powerful architectures with wider word length, more execution units and so on – a relatively simple way to achieve software compatibility between different products in a processor family.

Some hardware vendors, especially IBM, use the term "microcode" as a synonym for "firmware". That way, all code in a device is termed "microcode" regardless of it being microcode or machine code; for example, hard disk drives are said to have their microcode updated, though they typically contain both microcode and firmware.

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	With a neat block diagram explain the working principle of micro program sequencer	Dec 2010	5
Q.2	Draw the format of a microinstruction and explain how a microprogram sequencer works.	June 2010	7
Q.3	Write a brief note on microprogram sequencer	June 2014	2

Unit-02/Lecture-05

Arithmetic Logic Unit

Arithmetic Logic Unit, ALU is one of the many components within a computer processor. The ALU performs mathematical, logical, and decision operations in a computer and is the final processing performed by the processor. After the information has been processed by the ALU, it is sent to the computer memory.

In some computer processors, the ALU is divided into an AU and LU. The AU performs the arithmetic operations and the LU performs the logical operations.

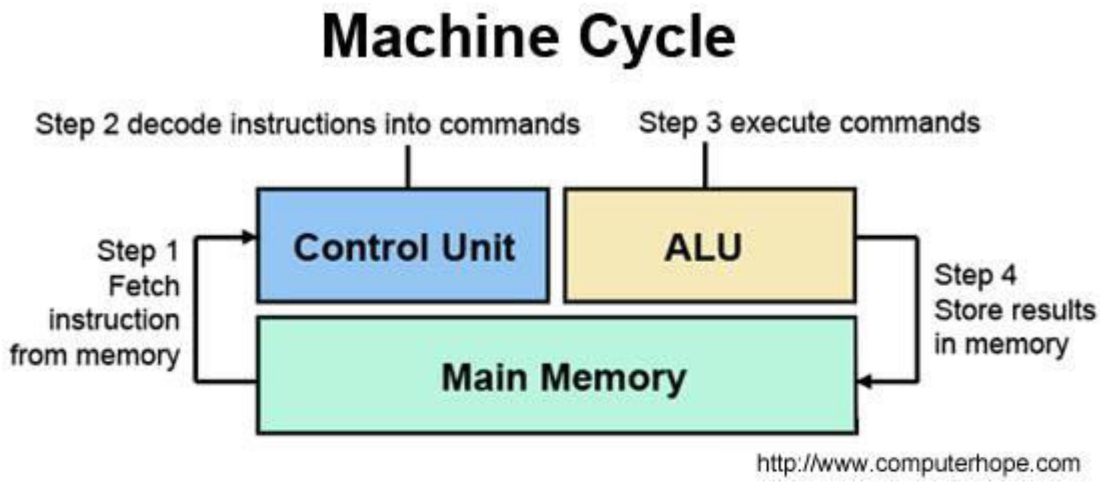


Fig 2.5 Arithmetic Logic Unit

An arithmetic logic unit (ALU) is a digital circuit that performs integer arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit of a computer, and even the simplest microprocessors contain one for purposes such as maintaining timers. The processors found inside modern CPUs and graphics processing units (GPUs) accommodate very powerful and very complex ALUs; a single component may contain a number of ALUs.

An ALU must process numbers using the same formats as the rest of the digital circuit. The format of modern processors is almost always the two's complement binary number representation. Early computers used a wide variety of number systems, including ones' complement, two's complement, sign-magnitude format, and even true decimal systems, with various representation of the digits.

The ones' complement and two's complement number systems allow for subtraction to be accomplished by adding the negative of a number in a very simple way which negates the need for specialized circuits to do subtraction; however, calculating the negative in two's complement requires adding a one to the low order bit and propagating the carry. An alternative way to do two's complement subtraction of A−B is to present a one to the carry input of the adder and use ¬B rather than B as the second input. The arithmetic, logic and shift circuits introduced in previous sections can be combined into one ALU with common selection

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU). Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU and memory.

ALU working

An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

All information in a computer is stored and manipulated in the form of binary numbers, i.e. 0 and 1. Transistor switches are used to manipulate binary numbers, since there are only two possible states of a switch: open or closed. An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1. Operations can be accomplished by connecting multiple transistors. One transistor can be used to control a second one, in effect turning the transistor switch on or off depending on the state of the second transistor. This is referred to as a gate, because the arrangement can be used to allow or stop a current. The simplest type of operation is a NOT gate. This uses only a single transistor. It uses a single input and produces a single output, which is always the opposite of the input. The figure below shows the logic of the NOT gate.

Unit-03/Lecture-06

These computational units process the binary data directly and provide support for arithmetic computations of varying precision. The ALU performs a standard set of arithmetic and logic operations as well as division. The MAC performs multiply, multiply/add, and multiply/subtract operations. The shifter performs logical and arithmetic shifts, normalization, denormalization, and other operations.

Addition and Subtraction

Addition is the most common arithmetic operation a processor performs. When two n-bit numbers are added together, it is always possible to produce a result with n + 1 nonzero digits due to a carry from the leftmost digit. For two's complement addition of two numbers, there are three cases to consider:

- If both numbers are positive and the result of their addition has a sign bit of 1, then overflow has occurred; otherwise the result is correct.
- If both numbers are negative and the sign of the result is 0, then overflow has occurred; otherwise the result is correct.
- If the numbers are of unlike sign, overflow cannot occur and the result is always correct.
- For addition use normal binary addition
- 0+0=sum 0 carry 0
- 0+1=sum 1 carry 0
- 1+1=sum 0 carry 1

Monitor MSB for overflow

Overflow cannot occur when adding 2 operands with the different signs. If 2 operand have same sign and result has a different sign, overflow has occurred. Subtraction: Take 2’s complement of subtrahend and add to minuend i.e. a -b = a + (b),So we only need addition and complement circuits

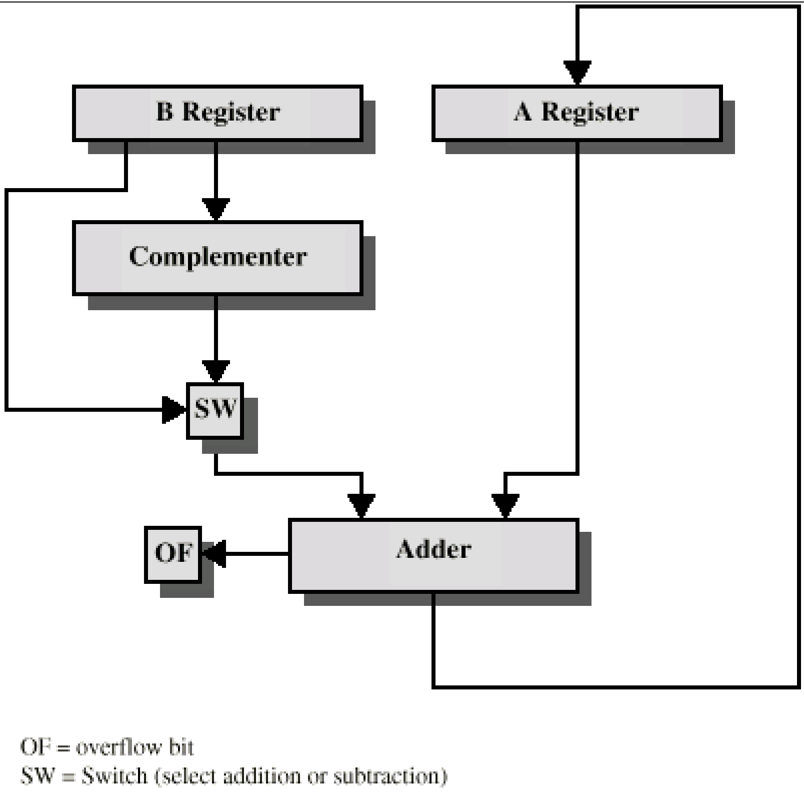


Fig. 2.6 Hardware for Addition & Subtraction

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Draw and explain flowchart for addition and subtraction of floating point number	June 2009, June 2012	7
Q.2	Write down the algorithm for addition and subtraction with signed-magnitude data. Also draw the flowchart.	June 2011	10

Unit-02/Lecture-07

Multiplication

A complex operation compared with addition and subtraction. Many algorithms are used, esp. for large numbers .Simple algorithm is the same long multiplication taught in grade school.

Compute partial product for each digit. Add partial products.

Multiplication Example

•1011 Multiplicand (11 dec)

•x 1101 Multiplier (13 dec)

•1011 Partial products

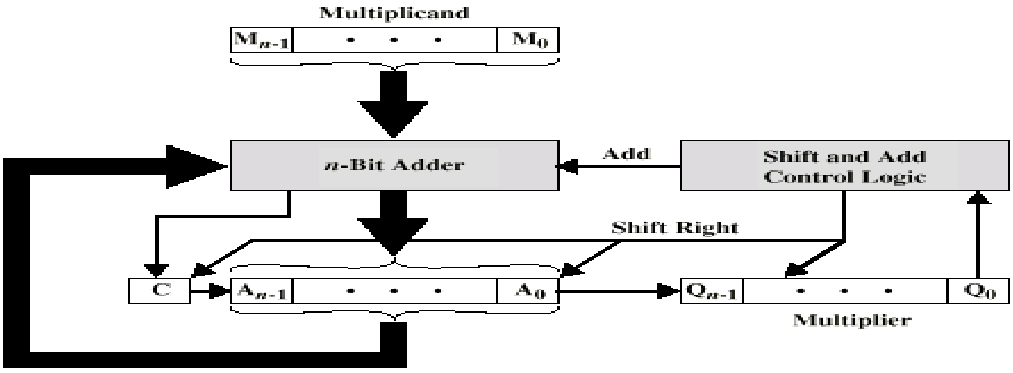
•000 Note: if multiplier bit is 1 copy

•1011multiplicand (place value)

•1011otherwise zero

•10001111 Product (143 dec)

•Note: need double length result



(a) Block Diagram

Fig 2.7

Multiplication Algorithm

Repeat n times:

If Q0= 1 Add M into A, store carry in CF

Shift CF, A, Q right one bit so that:

An-<CFQn-1 <-A0–Q0 is lost

Note that during execution Q contains bits from both product and multiplier

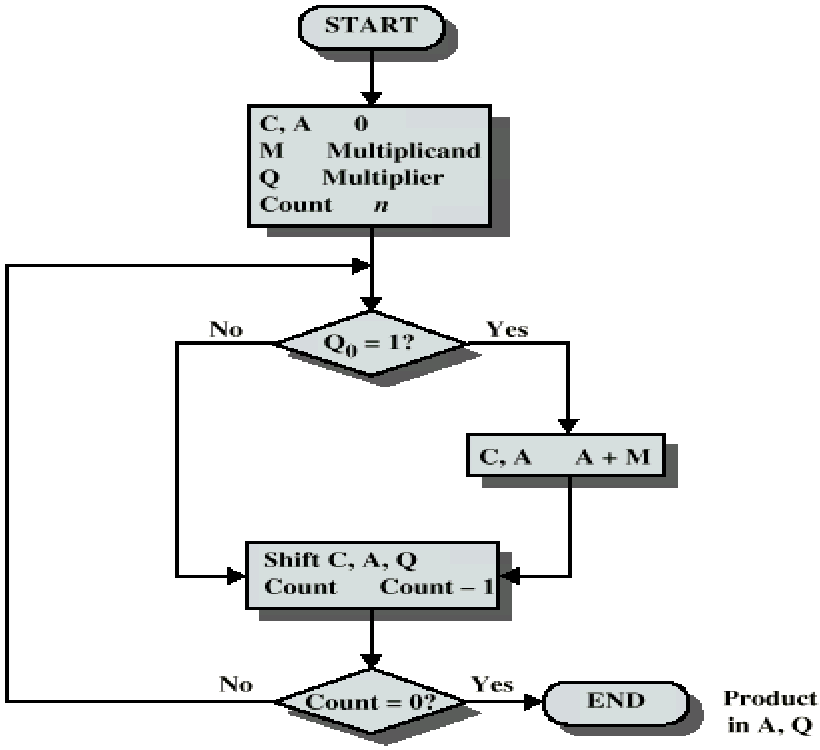


Fig 2.8 Flowchart for Unsigned Binary Multiplication

Division

•More complex than multiplication to implement (for computers as well as humans. Some processors designed for embedded applications or digital signal processing lack a divide instruction.

•Basically inverse of add and shift: shift and subtract.

Unsigned Division algorithm
• Using same registers (A,M,Q, count) as multiplication
• Results of division are quotient and remainder
Q will hold the quotient
A will hold the remainder
• Initial values
Q <- 0
A <- Dividend
M <- Divisor
Count <- n

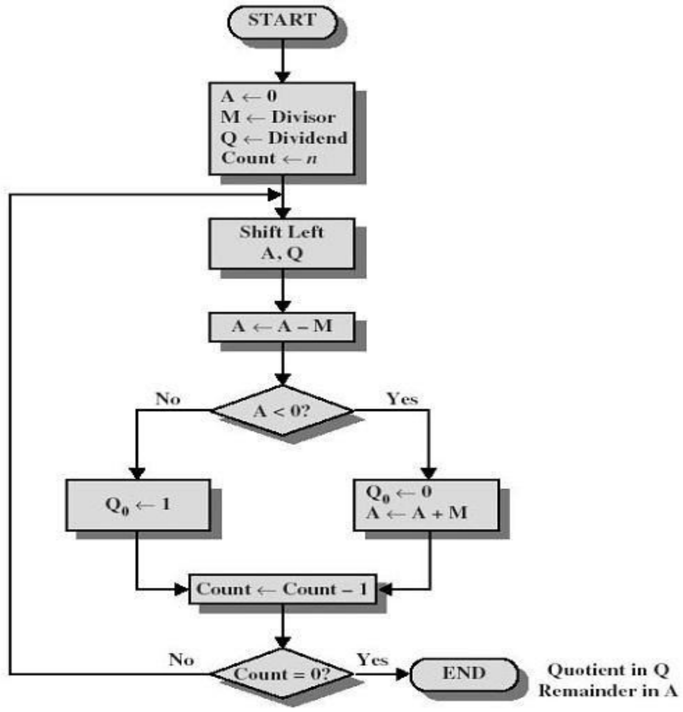


Fig 2.9 Flowchart for Division

Unit-02/Lecture-08

Floating Point Arithmetic

Arithmetic operations on floating point numbers consist of addition, subtraction, multiplication and division The operations are done with algorithms similar to those used on sign magnitude integers (because of the similarity of representation) -- example, only add numbers of the same sign. If the numbers are of opposite sign, must do subtraction.

Arithmetic unit

The arithmetic unit, also called the arithmetic logic unit (ALU), is a component of the central processing unit (CPU). It is often referred to as the “engine” of the CPU because it allows the computer to perform mathematical calculations, such as addition, subtraction, and multiplication. The ALU also performs logic operations, like “AND,” “OR,” and “NOT.” The arithmetic unit works along with the register array, which holds data, when processing any of these operations. The arithmetic unit is comprised of many interconnected elements that are designed to perform specific tasks.

Some central processing units are comprised of two components, an arithmetic unit and a logic unit. Other processors may have an arithmetic unit for calculating fixed-point operations and another AU for calculating floating-point computations. Some PCs have a separate chip known as the numeric coprocessor. This coprocessor contains a floating-point unit for processing floating-point operands. The coprocessor increases the operating speed of the computer because of the coprocessor ability to perform computation faster and more efficiently.

	RGPV QUESTIONS	Year	Marks
Q.1	Take an example and explain the design of arithmetic and logic unit	June 2014	7

Unit-02/Lecture-09

An Arithmetic Unit is a combinational circuit that performs arithmetic microoperations on a pair of n-bit operands (ex. A[3:0] and B[3:0]). The operations performed by an AU are controlled by a set of function-select inputs. In this design a 4-bit AU with 2 function-select inputs: Select S1 and S0 inputs. The functions performed by the AU are specified in Table A block diagram is given in Figure

S1	S0	C0	FUNCTION	OPERATION
0	0	0	A	Transfer A
0	0	1	A + 1	Increment A by 1
0	1	0	A + B	Add A and B
0	1	1	A + B + 1	Increment the sum of A and B by 1
1	0	0	A + B'	A plus one's complement of B
1	0	1	A – B	Subtract B from A (i.e. B' + A + 1)
1	1	0	A' + B	B plus one's complement of A
1	1	1	B – A	B minus A (or A' + B + 1)

Table No.2.1

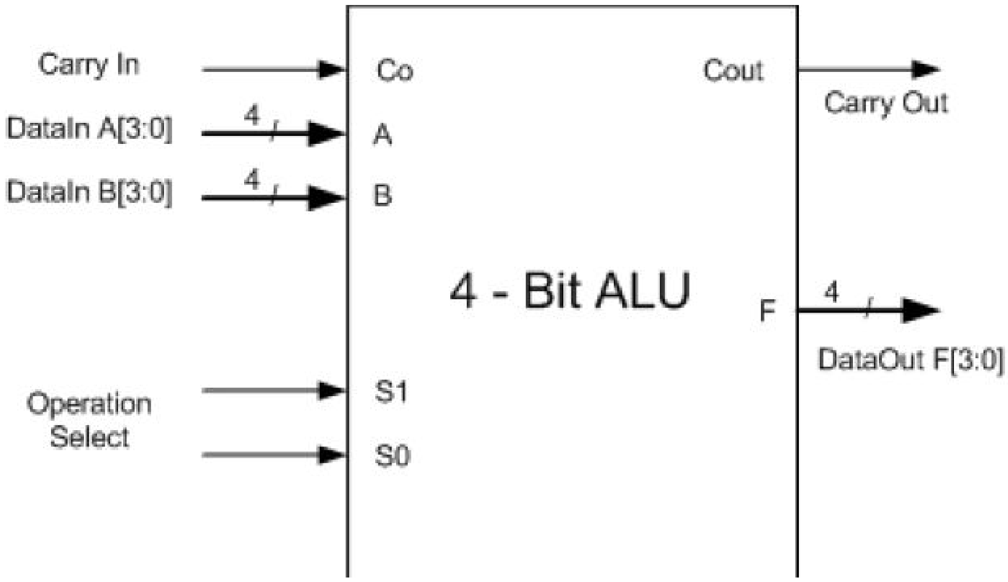


Fig 2. 10 Block Diagram of Arithmetic Unit

We will be using two's complement system of notation while dealing with arithmetic operations in our AU. This has a number of advantages over the sign and magnitude representation such as easy addition or subtraction of mixed positive and negative numbers. Recall that the two's complement of a n-bit number N is defined as:

$2^n - N = (2^n - 1 - N) + 1$

The last representation gives us an easy way to find two's complement: take the bit wise complement of the number and add 1 to it. As an example, to represent the number -5, we take two's complement of 5 (=0101) as follows,

5 0 1 0 1 --> 1 0 1 0 (bit wise complement)

+ 1

1 0 1 1 (two's complement)

Numbers represented in two's complement lie within the range **$-(2^{n-1})$** to + **$(2^{n-1} - 1)$**. For a 4-bit number this means that the number is in the range of -8 to +7. There is a potential problem we still need to be aware of when working with two's complement, namely overflow and underflow as is illustrated in the examples below,

0 1 0 0 (=carry Ci)

+5 0 1 0 1

+4 + 0 1 0 0

+9 0 1 0 0 1 = -7!

Also,

1 0 0 0 (=carry Ci)

-7 1 0 0 1

-2 + 1 1 1 0

-9 1 0 1 1 1 = +7!

Both calculations give the wrong results (-7 instead of +9 or +7 instead of -9) which is caused by the fact that the result +9 or -9 is out of the allowable range for a 4- bit two’s complement number. Whenever the result is larger than +7 or smaller than -8 there is an overflow or underflow and the result of the addition or subtraction is wrong. Overflow and underflow can be easily detected when the carry out of the most significant stage (i.e. C_4) is different from the carry out of the previous stage (i.e. C_3).

The inputs ***A and B have to be presented in two’s complement to the inputs of the AU***.

	RGPV QUESTIONS	Year	Marks
Q.1	Take an example and explain the design of arithmetic and logic unit	June 2014	7