BCA Part I

Paper-VII

Topic: ALU and Control Unit

Prepared by: Dr. Kiran Pandey
(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

ALU Organisation

ALU is a fundamental building block of the central processing unit (CPU) found in many computers. It is a digital circuit that performs simple arithmetic-logic and shift operations. The complexity of an ALU is determined by the way in which its arithmetic instructions are realized for it. The simple ALUs can be constructed using combinational circuits for fixed-point numbers that perform addition and subtraction as well as word based logical operations. On the other hand the floating-point arithmetic implementation requires more complex control logic and data processing capabilities, i.e., the hardware. Several micro-processor families utilize only fixed-point arithmetic capabilities in the ALUs. Some processors having fixed point ALU's have special purpose auxiliary unit called arithmetic co-processor. We will discuss all these issues in greater detail in this section.

A Simple ALU Organisation

An ALU consists of circuits that perform data processing micro-operations. The simplest organisation in this respect for fixed point ALU was suggested by John von Neumann in his IAS computer design. Figure I shows the widely used sequential ALU that aims at minimizing hardware cost.

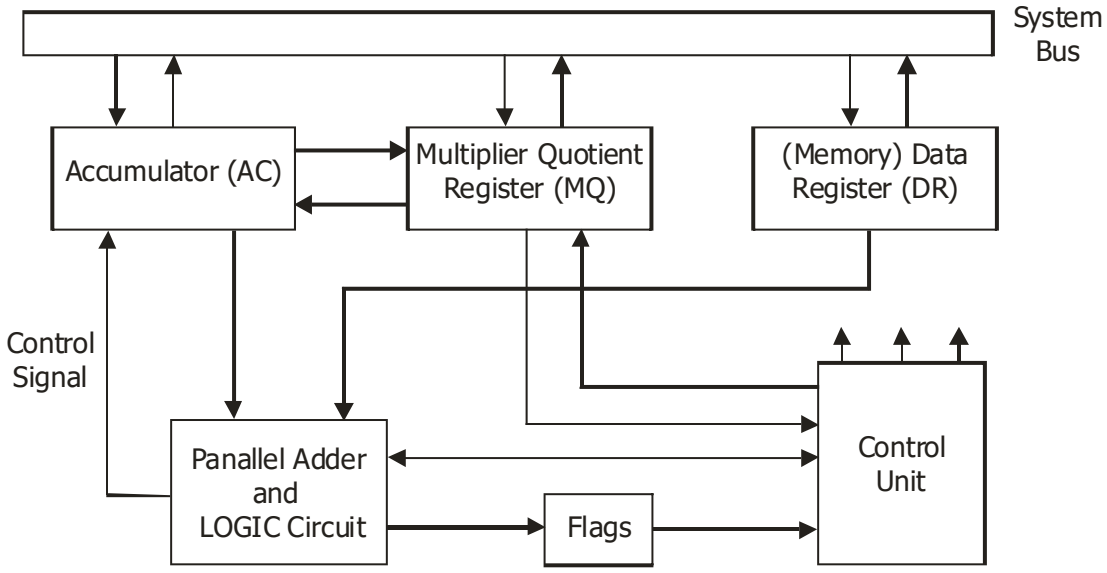


Figure 1 : Structure of a Sequential ALU

The above structure has three registers : the accumulator **AC**, the multiplier quotient **MQ** and the data register **DR** for data storage. We will assume that they are equal to one word each. In the figure above we observe that any ALU operation at most can have two input values and will generate single output along with the other status bits. In the present case the two inputs are **AC** and **DR** registers, while output is **AC** register. AC and MQ registers are generally used as a single AC. MQ register. This register is capable of left or right shift operations. Multiplication and division were implemented using shift-add/subtract operations. The MQ (Multiplier-Quotient register) is a special register used for implementation of multiplication and division. For multiplication or division operations DR register stores the multiplicand or divisor respectively. The result of multiplication or division (product, or quotient and remainder) on applying certain algorithm can finally be stored in AC.MQ register combination. Some of the micro-operations that can be defined on this ALU are :

Addition :	AC : = AC + DR
Subtraction :	AC : = AC – DR
Multiplication :	AC.MQ : = DR × MQ
Division :	AC.MQ : = MQ/DR
AND :	AC : = AC **and** DR
OR :	AC : = AC **v** DR
Exclusive OR :	AC : = AC **xor** DR
NOT :	AC : = **not** (AC)

DR is another important register, which is used as a memory data register for storing second operand. It stores data addressed by an instruction address field

ADR. Then DR can be replaced by M(ADR) in the above list of ALU operations, resulting in a one-address memory-referencing. In fact it acts as a buffer register, which stores the data brought from the memory for an instruction. In machines where we have general purpose registers any of the registers can be utilized as AC, MQ and DR.

Arithmetic Processors

A processor, if devoted exclusively to arithmetic functions, can be used to implement a full range of arithmetic functions in the hardware at a relatively low cost. This can be done in a single Integrated Circuit. Thus, a special purpose arithmetic processor can be constructed for performing only the arithmetic operations. This processor physically may be separate, yet can be utilized by the CPU to execute complex arithmetic instructions. Please note in the absence of arithmetic processors, These instructions may be executed using the slower software routines by the CPU itself. This auxiliary processor enhances the speed of execution of programs having a lot of complex arithmetic computations.

An arithmetic processor also helps in reducing program complexity, as it provides a richer instruction set for a machine. Some of the instructions that can be assigned to arithmetic processors can be related to the addition, subtraction, multiplication, and division of floating point numbers, exponentiation, logarithms and other trigonometric functions.

Two mechanisms are used for connecting the arithmetic processor to the CPU.

(i) Loosely coupled

(ii) Tightly coupled

In loosely coupled an arithmetic processor is treated as one of the Input/Output or peripheral units. The CPU sends data and instructions to the peripheral processor, which performs the required operations on the data and communicates the results back to the CPU. A peripheral processor has several registers to communicate with the CPU. These registers may be addressed by the CPU as Input / Output register addresses. The CPU and peripheral processors are normally quite independent and communicate with each other by exchange of information using data transfer instructions. The data transfer instructions must be specific instructions in the CPU.

On the other hand a tightly coupled processor is an arithmetic processor that has a register and instruction set which is considered an extension of the CPU registers and instruction set. Here the CPU reserves a special subset of code for arithmetic processor. In such a system the instructions meant for arithmetic processor are fetched by CPU and decoded jointly by CPU and the arithmetic processor, and finally executed by arithmetic processor. Thus, these processors can

be considered a logical extension of the CPU. Such attached arithmetic processors are termed as **co-processors**. An arithmetic **coprocessor** is a computer processor used to supplement the functions of the primary processor (the CPU). Operations performed by the coprocessor may be floating point arithmetic, graphics, signal processing, string processing, encryption or I/O Interfacing with peripheral devices. By offloading processor-intensive tasks from the main processor, coprocessors can accelerate system performance. Coprocessors allow a line of computers to be customized, so that customers who do not need the extra performance don't need to pay for it.

The concept of co-processor existed in the 8086 machine till Intel 486 machines where co-processor was separate. However, Pentium at present does not have a separate co-processor. Similarly, peripheral processors are not found as arithmetic processors in general. However, many chips are used for specialized I/O architecture.

The Control Unit

The control unit of the CPU selects and interprets program instructions and then sees that they are executed. The basic responsibilities of the control unit are **to control** :

(i) Data exchange of CPU with the memory or I/O modules.

(ii) Internal operations in the CPU such as :

- moving data between registers (register transfer operations)
- making ALU to perform a particular operation on the data
- regulating other internal operations.

Functions of Control Unit

To define the functions of a control unit, we must know what resources and means it has at its disposal. A control unit must know about the basic components of the CPU and Micro-operation the CPU performs.

The CPU of a computer consists of the following basic functional components :

- **The Arithmetic Logic Unit (ALU)–**Performs the basic arithmetic and logical operations.
- **Registers–**are used for information storage within the CPU.
- **Internal Data Paths–**These paths are useful for moving the data between two registers or between a register and ALU.

- **External Data Paths–**The roles of these data paths are normally to link the CPU registers with the memory or I/O interfaces. This role is normally fulfilled by the system bus.
- **The Control Unit–**This causes all the operations to happen in the CPU.

The micro-operations performed by the CPU can be classified as :

- Micro-operations for data transfer from register-register, register-memory, I/O-register etc.
- Micro-operations for performing arithmetic, logic and shift operations. These micro-operations involve use of registers for input and output.

The basic responsibility of the control unit is that it must be able to guide the various components of CPU to perform a specific sequence of micro-operations to achieve the execution of an instruction. The instruction execution is achieved by executing micro-operations in a specific sequence. For different instructions this sequence may be different. Thus the control unit must perform two basic functions :

- Cause the execution of a micro-operation.
- Enable the CPU to execute a proper sequence of micro-operations, which is determined by the instruction to be executed.

To achieve this task the control unit generates control signals

Structure of Control Unit

A control unit has a set of input values on the basic of which it produces an output control signal, which in turn performs micro-operations. These output signals control the execution of a program. A general model of control unit is shown in Fig. 3.

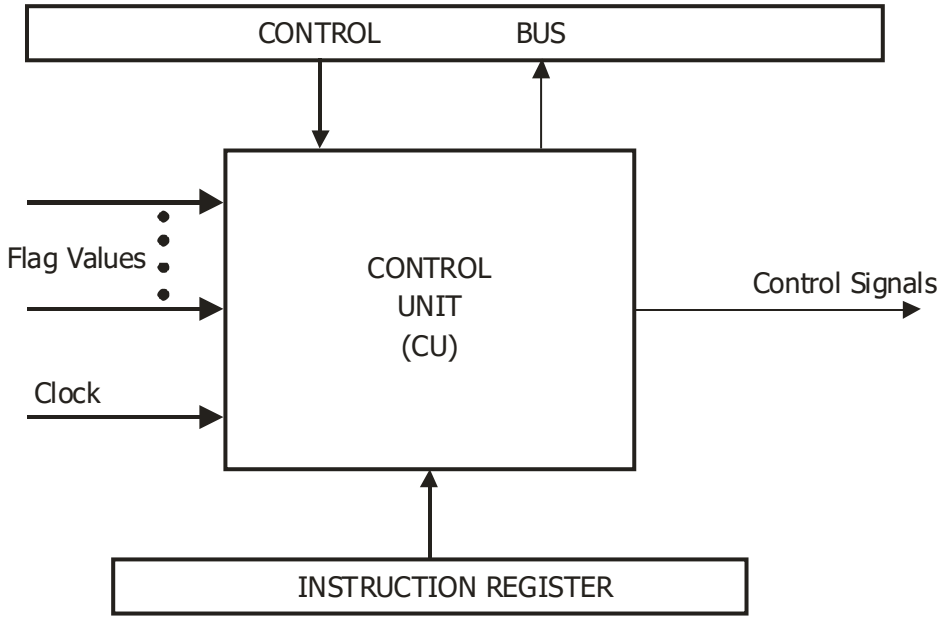


Figure 2 : A General Model of Control Unit

The inputs to the control unit are :

- **The Master Clock Signal :** This signal causes micro-operations to be performed in a square. In a single clock cycle either a single or a set of simultaneous micro-operations can be performed. The time taken in performing a single micro-operation is also termed as processor cycle time or the clock cycle time in some machines.
- **The Instruction Register :** It contains the operation code (opcode) and addressing mode bits of the instruction. It helps in determining the various cycles to be performed and hence determines the related micro-operations, which are needed to be performed.
- **Flags :** Flags are used by the control unit for determining the status of the CPU and the outcomes of a previous ALU operation. For example, a zero flag if set, conveys to control unit that for instruction ISZ (skip the next instruction if zero flag is set) the next instruction is to be skipped. For such a case control unit cause increment of PC by program instruction length, thus skipping next instruction.
- **Control Signals from Control Bus :** Some of the control signals are provided to the control unit through the control bus. These signals are issued from outside the CPU. Some of these signals are interrupt signals and acknowledgement signals.

On the basis of the input signals the control unit activates certain output control signals, which in turn are responsible for the execution of an instruction. These output control signals are :

- **Control signals, which are required within the CPU :** These control signals cause two types of micro-operations, viz., for data transfer from one register to another; and for performing an arithmetic, logic and shift operation using ALU.
- **Control signals to control bus :**These control signals transfer data from or to CPU register to or from memory or I/O interface. These control signals are issued on the control bus to activate a data path on the data/address bus etc.

A prime requirement for control unit is that it must know how all the instructions will be executed. It should also know about the nature of the results and the indication of possible errors. All this is achieved with the help of flags, op-codes, clock and some control signals to itself.

A control unit contains a clock portion that provides clock-pulses. This clock signal is used for measuring the timing of the micro-operations. In general, the timing signals from control unit are kept sufficiently long to accommodate the

proportional delays of signals within the CPU along various data paths. Since within the same instruction cycle different control signals are generated at different times for performing different micro-operations, therefore a counter can be utilized with the clock to keep the count. However, at the end of each instruction cycle the counter should be reset to the initial condition. Thus, the clock to the control unit must provide counted timing signals.

To achieve a particular operation the control signals are applied directly as the binary inputs to the logic gates of the logic circuits. All these inputs are the control signals, which are applied to select a circuit (for example, select or enable input) or a path (for example, multiplexers) or any other operation in the logic circuits.

A program execution consists of a sequence of instruction cycles. Each instruction cycle is made up of a number of sub cycles. One such simple subdivision includes fetch, indirect, execute, and interrupt cycles, with only fetch and execute cycles always occurring. Each sub cycle involves one or more micro-operations.

Let us take an example to discuss how the events of any instruction cycle can be described as a sequence of such micro-operations.

The Fetch Cycle

The Fetch Cycle requires four Registers :

- **Memory Address Register (MAR)–** Connected to address bus and specifies address for read or write operation
- **Memory Buffer Register (MBR)–** Connected to data bus and holds data to write or last data read
- **Program Counter (PC)–** Holds address of next instruction to be fetched
- **Instruction Register (IR)–** Holds last instruction fetched

Fetch sequence is in the following way :

- **Address of next instruction is in PC –** Address (MAR) is placed on address bus and the Control unit issues READ command.
- **Result (data from memory) appears on data bus –** Data from data bus is copied into MBR and PC is incremented by instruction length (in parallel with data fetch from memory).
- **Data (Instruction) moved from MBR to IR –** MBR is now free for further data fetch.

The fetch cycle actually consists of 3 step and 4 micro operations. Each micro-operations consists of moving data in or out of a register. Those micro-operations

that do not conflict can be executed in parallel. The fetch sequence can be given as follows :

- t1 : MAR ← (PC)
- t2 : MBR ← (memory)
  PC ← (PC) + 1
- t3 : IR ← (MBR)
  (tx = time unit/clock cycle) or
- t1 : MAR ← (PC)
- t2 : MBR ← (memory)
- t3 : PC ← (PC) + 1
  IR ← (MBR)

The Indirect Cycle

Once an instruction is fetched, the next step is to fetch the operands. Considering the same example as of an indirect address is handled using indirect cycle. The following micro-operations are required in the indirect cycle :

- t1 : MAR ← (IR address) - address field of IR
- t2 : MBR ← (memory)
- t3 : IR address ← (MBR address)

Now MBR contains direct address of operand and IR is updated with direct address of operand. IR is now in same state as if direct addressing had been used. The MAR is loaded with the address field of IR register. Then the memory is read to fetch the address of operand, which is transferred to the address field of IR through MBR as data is received in MBR during the read operation. IR is now ready for the execute cycle.

The Execute Cycle

The fetch and indirect cycles involve a small, fixed sequence of micro-operations. Each of these cycles has fixed sequence of micro-operations that are common to all instructions.

This is not true of the execute cycle. For a machine with N different opcodes, there are N different sequences of micro-operations that can occur. Let us consider some hypothetical instruction :

An add instruction that adds the contents of memory location X to Register R1 with R1 storing the result :

ADD R1, X

The sequence of micro-operations may be :

t1 : MAR ← IR (address)

t2 : MBR ← [MAR]

t3 : R1 ← R1 + MBR

At the beginning of the execute cycle IR contains the ADD instruction and its direct operand address (memory location X). At time t_1, the address portion of the IR is transferred to the MAR. At t_2 the referenced memory location is read into MBR Finally, at t_3 the contents of R1 and MBR are added by the ALU.

The Interrupt Cycle

At end of execute cycle, processor tests interrupt signal. If so, the interrupt cycle is performed. This cycle does not execute an interrupt but causes start of execution of Interrupt Service Program (ISR). Please note that ISR is executed as just another program instruction cycle. The nature of this cycle varies greatly from one machine to another. A typical sequence of micro-operations of the interrupt cycle are :

t1 : MBR ← (PC)

t2 : MAR ← save-address PC ← routine-address

t3 : memory ← (MBR)

Most processors provide multiple types of address and so thee may be additional micro-ops to get addresses. At time t_1, the contents of the PC are transferred to the MBR, so that they can be saved for return from the interrupt. At time t_2 the MAR is loaded with the address at which the contents of the PC are to be saved, and PC is loaded with the address of the start of the interrupt-servicing routine. At time t_3 MBR, which contains the old value of the PC, is stored in the memory. The processor is now ready to begin the next instruction cycle.

The Instruction Cycle

The instruction cycle for this given machine consists of four cycles. Let us assume a 2-bit Instruction Cycle Code (ICC). The ICC can represent the state of the processor in terms of cycle. For example, we can use:

00 : Fetch

01 : Indirect

10 : Execute

11 : Interrupt

At the end of each of the four cycles, the ICC is set appropriately. Please note that an indirect cycle is always followed by the execute cycle and the interrupt cycle

is always followed by the fetch cycle. For both the execute and fetch cycles, the next cycle depends on the state of the system.

The Hardwired Control

There are two major categories of control unit organisation :

1. Hardwired control organization
2. Micro programmed control organization

In the hardwired organization, the control unit is designed as a combinational circuit. That is, the control unit is implemented by gates, flip-flops, decoder and other digital circuits. Hardwired control units can be optimised for fast operations.

The block diagram of control unit is shown in Figure. The major inputs to the circuit are instruction register, the clock, and the flags. The control unit uses the opcode of instruction stored in the IR register to perform different actions for different instructions. The control unit logic has a unique logic input for each opcode. This simplifies the control logic. This control line selection can be performed by a decoder.

A decoder will have n binary inputs and 2^n binary outputs. Each of these 2^n different input patterns will activate a single unique output line.

The clock portion of the control unit issues a repetitive sequence of pulses for the SS duration of micro-operation(s). These timing signals control the sequence of execution of instruction and determine what control signal needs to applied at what time for instruction execution.

Wilkes Control

Prof. M.V. Wilkes of the Cambridge University Mathematical Laboratory coined the term microprogramming in 1951. He provided a systematic alternative procedure for designing the control unit of a digital computer. During instruction executing a machine instruction, a sequence of transformations and transfer of information from one register in the processor to another take place. These were also called the micro operations. **Because of the analogy between the execution of individual steps in a machine instruction to the execution of the individual instruction in a program, Wilkes introduced the concept of micro-programming.** The Wilkes control unit replaces the sequential and combinational circuits of hardwired control unit by a simple control unit in conjunction with a storage unit that stores the sequence of steps of instruction that is a micro-program.

In Wilkes microinstruction has two major components :

(a) Control field which indicates the control lines which are to be activated and

(b) Address field, which provides the address of the next microinstruction to be executed.

The Micro-Programmed Control

An alternative to a hardwired control unit is a micro-programmed control unit, in which the logic of the control unit is specified by a micro-program. A micro-program is also called firmware (midway between the hardware and the software). It consists of :

(a) One or more micro-operations to be executed; and

(b) The information about the micro-instruction to be executed next.

The micro-instructions are stored in the control memory. The address register for the control memory contains the address of the next instruction that is to be read. The control memory Buffer Register receives the micro-instruction that has been read. A micro-instruction execution primarily involves the generation of desired control signals and signals used to determine the next micro-instruction to be executed. The sequencing logic section loads the control memory address register. It also issues a read command to control memory. The following functions are performed by the micro-programmed control unit :

(i) The sequence logic unit specifies the address of the control memory word that is to be read, in the Address Register of the Control Memory. It also issues the READ signal.

(ii) The desired control memory word is read into control memory Buffer Register.

(iii) The content of the control memory buffer register is decoded to create control signals and next-address information for the sequencing logic unit.

(iv) The sequencing logic unit finds the address of the next control word on the basis of the next-address information from the decoder and the ALU flags.

The execute cycle steps of micro-operations are different for all instructions in addition the addressing mode may be different. All such information generally is dependent on the opcode of the instruction Register (IR). Thus, IR input to Address

Register for Control Memory is desirable. Thus, there exist a decoder from IR to Address Register for control memory. (Refer Figure). This decoder translates the opcode of the IR into a control memory address.

The Micro-Instructions

A micro-instruction, as defined earlier, is an instruction of a micro-program. It specifies one or more micro-operations, which can be executed simultaneously. On executing a micro-instruction a set of control signals are generated which in turn cause the desired micro-operation to happen.

Types of Micro-instructions

In general the micro-instruction can be categorised into two general types :

(i) branching

(ii) non-branching.

After execution of a non-branching micro-instruction the next micro-instruction is the one following the current micro-instruction. However, the sequences of micro-instructions are relatively small and last only for 3 or 4 micro-instructions. A conditional branching micro-instruction tests conditional variable or a flag generated by an ALU operation. Normally, the branch address is contained in the micro-instruction itself.

Control Memory Organization

One of the simplest ways to organize control memory is to arrange micro-instructions for various sub cycles of the machine instruction in the memory. The Figure 3 shows such an organisation.

Micro-Instruction for fetch cycle.	Fetch cycle
Micro-operation for Indirect cycle.	Indirect cycle
Micro-operation for interrupt initiation jump to fetch cycle.	Interrupt cycle
Micro instructions for calculating.	Execute cycle
(LOAD) Jump to fetch or interrupt cycle.	
(ADD) Jump to fetch or interrupt cycle.	
(ISZ) Jump to fetch or interrupt cycle.	

Figure 3 : Control Memory Organisation

For example let us take a machine instruction : **Branch on zero**

This instruction causes a branch to a specified main memory address in case the result of the last ALU operation is zero, that is, the zero flag is set. The pseudo code of the micro-program for this instruction can be written as :

Test"” If SET branch to label ZERO

Unconditional branch to label NON-ZERO

ZERO : (Microcode which causes replacement of program counter with the address provided in the instruction)

Branch to interrupt or fetch cycle.

NON-ZERO : (Microcode which may set flags if desired indicating the branch has not taken place).

Branch to interrupt or fetch. (For Next-Instruction Cycle)

Micro-instruction Formats

The two widely used formats used for micro-instructions are :

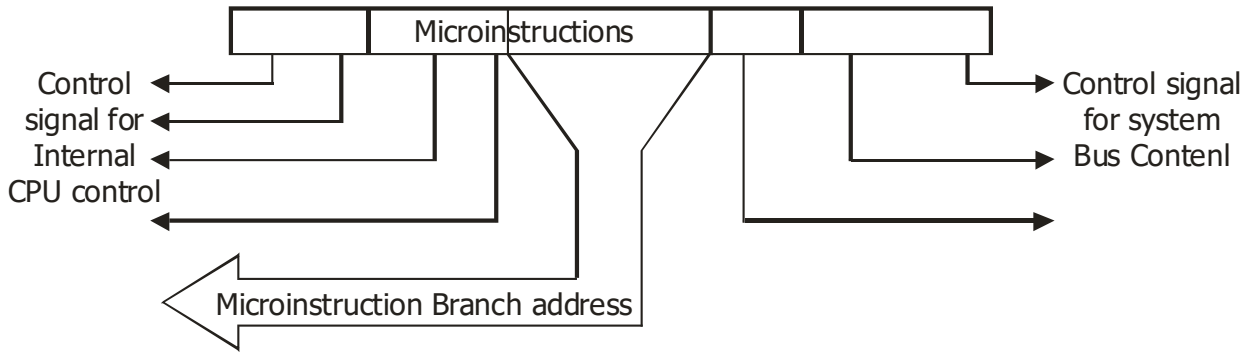
(i) Horizontal micro-instructions

(ii) Vertical micro-instructions

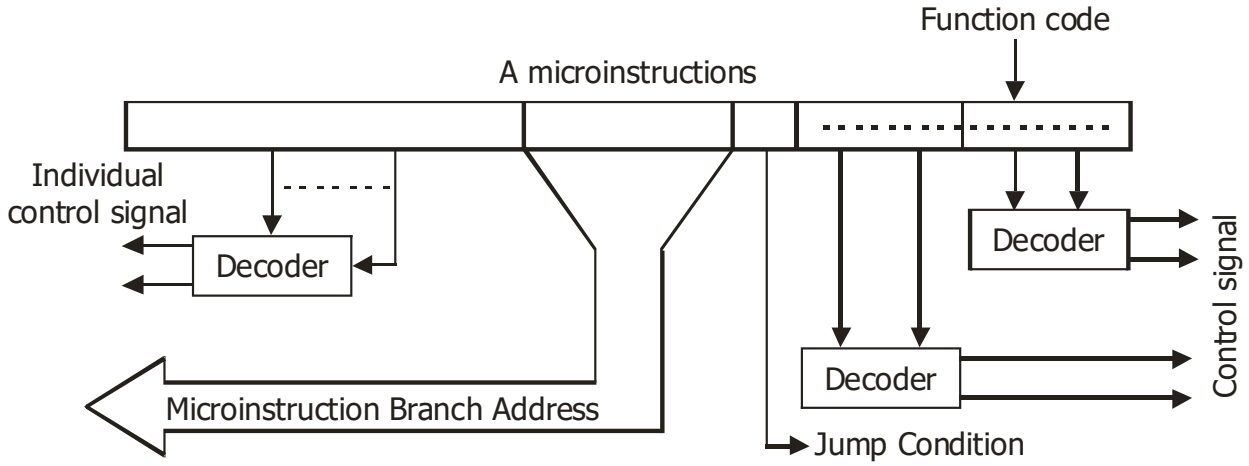
Horizontal micro-instruction have the following general attributes :

- Long format, may be hundred of bits.
- Ability to express high degree of parallelism.
- Little encoding of control information.

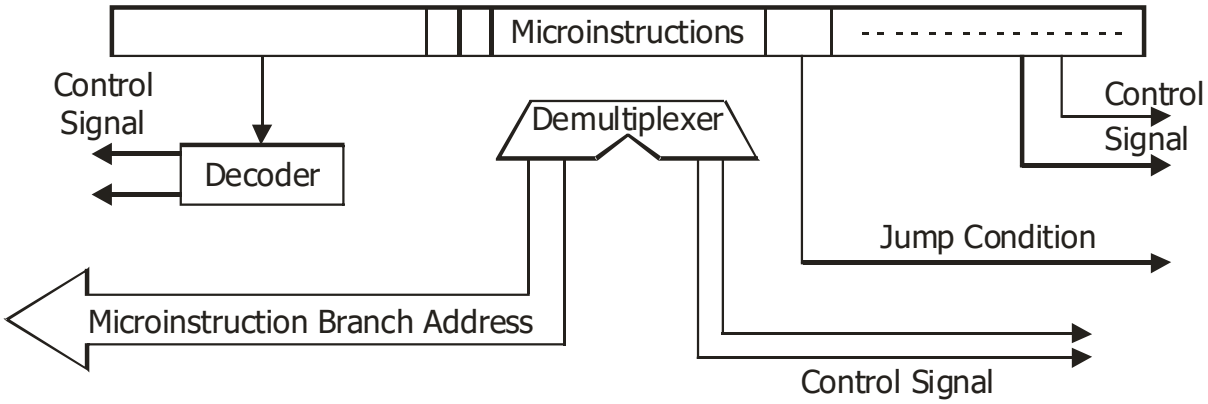
A typical horizontal micro-instruction with its related fields is shown in Figure (a).



(a) Horizontal Micro-instruction



(b) Vertical Micro-instructions



(c) A Realistic Micro-instructions

Figure 4 : Micro-instruction Formats

Vertical micro-instruction are characterised by :

- Short formats
- limited ability to express parallel micro-operation
- Considerable encoding of the control information.

In general, a horizontal control unit is faster, yet requires wider instruction words, whereas vertical control units, although; require a decoder, are shorter in length. Most of the systems use neither purely horizontal nor purely vertical micro-instructions figure 4 (c).

Questions

1. Describe ALU organization with diagram.
2. Define Arithmetic Processor. Why it is used ?
3. What is control unit ? Explain functional requirement of control unit.

4. What are the inputs to control unit.
5. Describe different types of control unit with diagram.
6. What are the two major components of Wilkes microinstruction.
7. How many address fields are there in Wilkes control unit ?
8. Differentiate between Direct Encoding and Indirect Encoding.