UNIT 4 THE CONTROL UNIT

Structure **Page No.**

4.0 INTRODUCTION

By now we have discussed instruction sets and register organisation followed by a discussion on micro-operations and a simple arithmetic logic unit circuit. We have also discussed the floating point ALU and arithmetic processors, which are commonly used for floating point computations.

In this unit we are going to discuss the functions of a control unit, its structure followed by the hardwired type of control unit. We will discuss the micro-programmed control unit, which are quite popular in modern computers because of flexibility in designing. We will start the discussion with several definitions about the unit followed by Wilkes control unit. Finally, we will discuss the concepts involved in micro-instruction execution.

4.1 OBJECTIVES

After going through this unit you will be able to:

- define what is a control unit and its function;
- describe a simple control unit organization;
- define a hardwired control unit;
- define the micro-programmed control unit;
- define the term micro-instruction; and
- identify types and formats of micro-instruction.

4.2 THE CONTROL UNIT

The two basic components of a CPU are the control unit and the arithmetic and logic 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**:

a) Data exchange of CPU with the memory or I/O modules.
b) 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.

But how does a control unit control the above operations? What are the functional requirements of the control unit? What is its structure? Let us explore answers of these questions in the next sections.

Functional Requirements of a Control Unit

Let us first try to define the functions which a control unit must perform in order to get things to happen. But in order to define the functions of a control unit, one must know what resources and means it has at its disposal. A control unit must know about the:

(a) Basic components of the CPU

(b) Micro-operation this CPU performs.

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

- **The Arithmetic Logic Unit (ALU),** which performs the basic arithmetic and logical operations.
- **Registers** which 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 lies in the fact that the control unit 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.

What are the functions, which a control unit performs to make an instruction execution feasible? 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.

But how are these two tasks achieved? The control unit generates control signals, which in turn are responsible for achieving the above two tasks. But, how are these control signals generated? We will answer this question in later sections. First let us discuss a simple structure of control unit.

Structure of Control Unit

A control unit has a set of input values on the basis 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 Figure 1.

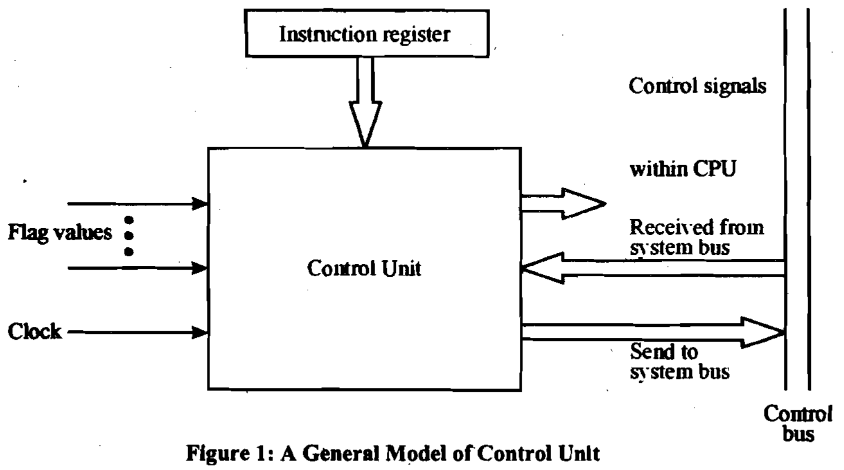


Figure 1: A General Model of Control Unit

In the model given above the control unit is a black box, which has certain inputs and outputs.

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 & 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.

Now, let us discuss the requirements from such a unit. 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 utilised 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. Examples, of the functionality of control units along with timing diagrams are given in further readings.

How are these control signals applied to achieve the partieular 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 revisit the micro-operations described in Unit 2 to discuss how the events of any instruction cycle can be described as a sequence of such micro-operations.

Fetch Cycle

The beginning of each instruction cycle is the fetch cycle, and causes an instruction to be fetched from memory.

The fetch cycle consists of four micro-operations that are executed in three timing steps. The fetch cycle can be written as:

T_1: MAR $\leftarrow$ PC
T_2: MBR $\leftarrow$ [MAR]
PC $\leftarrow$ PC + I
T_3: IR $\leftarrow$ MBR

where I is the instruction length. We assume that a clock is available for timing purposes and that it emits regularly spaced clock pulses. Each clock pulse defines a time unit. Thus, all the units are of equal duration. Each micro-operation can be performed within the time of a single time unit. The notation (T_1, T_2, T_3) represents successive time units. What is done in these time units?

- In the first time unit the content of PC is moved to MAR.

- In the second time unit the contents of memory location specified by MAR is moved to MBR and the contents of the PC is incremented by I.

- In the third time unit the content of MBR is moved to IR.

The Indirect Cycle

Once an instruction is fetched, the next step is to fetch the operands. Considering the same example as of Unit 2, the instruction may have direct and indirect addressing modes. An indirect address is handled using indirect cycle. The following micro-operations are required in the indirect cycle:

T_1 : MAR ← IR (address)
T_2 : MBR ← [MAR]
T_3 : IR (address) ← MBR (address)

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.

Thus, the IR now is in the same state as of direct address, viz., as if indirect addressing had not been used. 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 instructions:

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:

T_1 : MAR ← IR (address)

T_2 : MBR ← [MAR]

T_3 : 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.

Let us discuss one more instruction:

ISZ X, it increments the content of memory location X by 1. If the result is 0, the next instruction in the sequence is skipped. A possible sequence of micro-operations for this instruction may be:

T_1 : MAR ← IR (address)

T_2 : MBR ← [MAR]

T_3 : MBR ← MBR+ 1

T_4 : [MAR] ← MBR

If (MBR = 0) then (PC ← PC+ I)

Please note that for this machine we have assumed that MBR can be incremented by ALU directly.

The PC is incremented if MBR contains 0. This test and action can be implemented as one micro-operation. Note also that this micro-operation can be performed during the same time unit during which the updated value in MBR is stored back to memory. Such instructions are useful in implementing looping.

The Interrupt Cycle

On completion of the execute cycle the current instruction execution gets completed. At this point a test is made to determine whether any enabled interrupts have occurred. 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:

T_1: MBR ← PC

T_2: MAR ← Save-Address

PC ← ISR- Address

T_3: [MAR] ← MBR

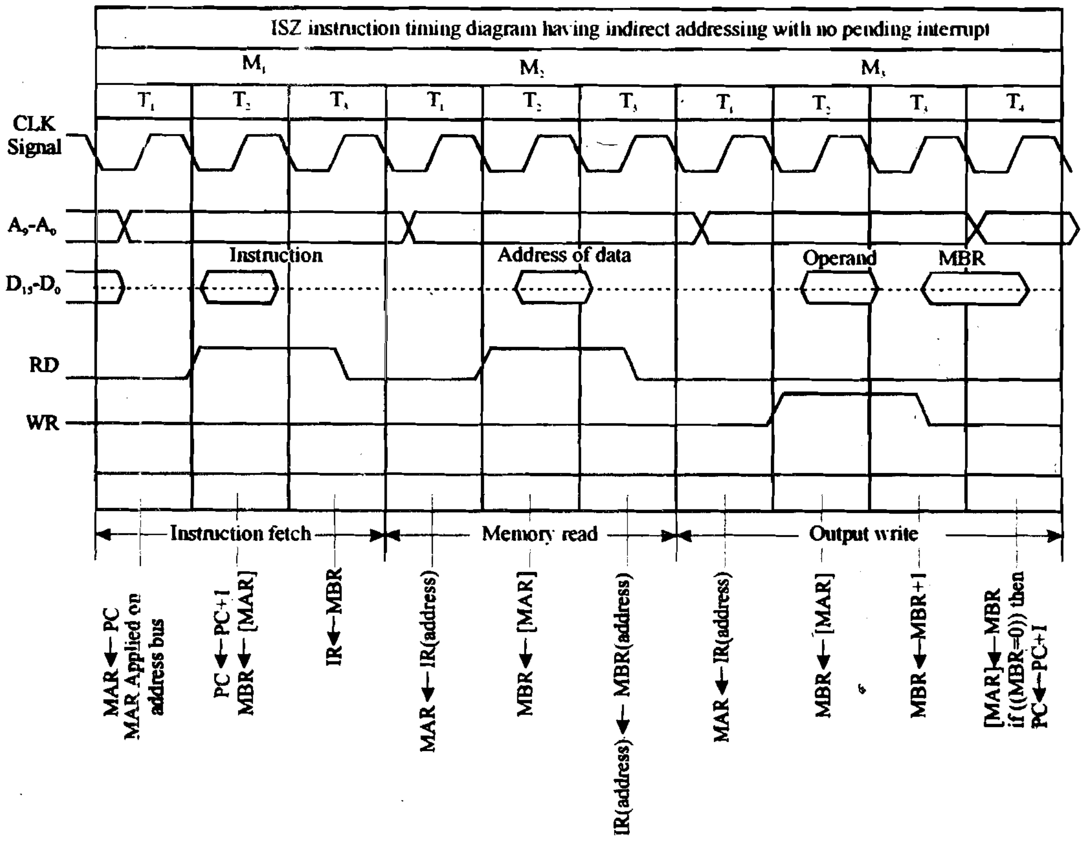
At time T_1, the contents of the PC are transferred to the MBR, so that they can be saved for return from the interrupt. At time T_2 the MAR is loaded with the address at which the contents of the PC are to be saved, and PC is loaded with the address of the start of the interrupt-servicing routine. At 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. 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. Let us show an instruction execution using timing diagram and instruction cycles:



Assumptions: 10 bit address bus, 16 bit data bus, size of instruction 16bits - with 10 bit address, 6 bit opcode

Figure 2: Timing Diagram for ISZ instruction

Please note that the address line determine the location of memory. Read/ write signal controls whether the data is being input or output. For example, at time T_2 in M_2 the read control signal becomes active, $A_9 - A_0$ input contains MAR that value is kept enabled on address bits and the data lines are enabled to accept data from RAM, thus enabling a typical RAM data output on the data bus.

For reading no data input is applied by CPU but it is put on data bus by memory after the read control signal to memory is activated. Write operation is activated along with data bus carrying the output value.

This diagram is used for illustration of timing and control. However, more information on these topics can be obtained from further readings.

4.3 THE HARDWIRED CONTROL

With the last section we have discussed the control unit in terms of its inputs, output and functions. A variety of techniques have been used to organize a control unit. Most of them fall into two major categories:

1. Hardwired control organization
2. Microprogrammed 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 3. 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.

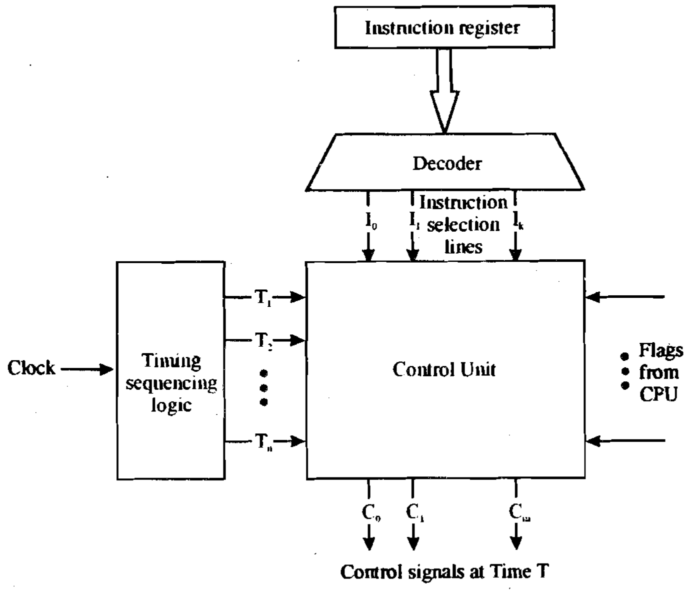


Figure 3: Block Diagram of Control Unit Operation

Check Your Progress 1

1. What are the inputs to control unit?

....

....

....

2. How does a control unit control the instruction cycle?

....

....

....

3. What is a hardwired control unit?

....

....

....

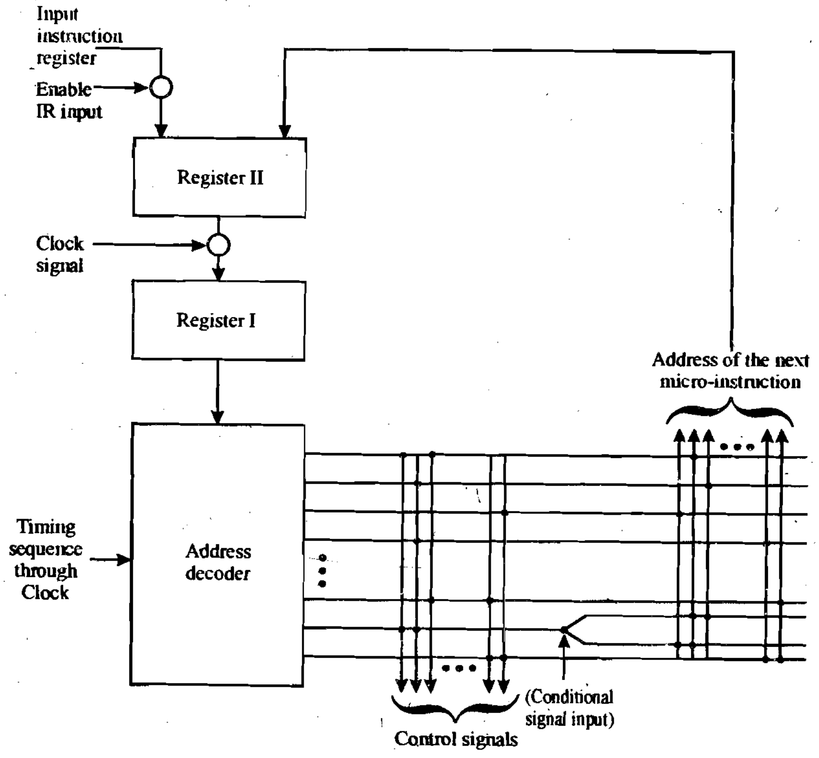
4.4 WILKES CONTROL

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 microprogramming.** 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 figure 4 below is an example of Wilkes control unit design.



Input instruction register
Enable IR input
Register II
Clock signal
Register I
Address of the next micro-instruction
Timing sequence through Clock
Address decoder
(Conditional signal input)
Control signals

Figure 4: Wilkes Control Unit

The control memory in Wilkes control is organized, as a PLA's like matrix made of diodes. This is partial matrix and consists of two components, the control signals and the address of the next micro-instruction. The register I contains the address of the next micro-instruction that is one step of instruction execution, for example T_1 in M_1 or T_2 in M_2 etc. as in Figure 2. On decoding the control signals are generated that cause execution of micro-operation(s) of that step. In addition, the control unit indicates the address of the next micro-operation which gets loaded through register II to register I. Register I can also be loaded by register II and "enable IR input" control signal. This will pass the address of first micro-instruction of execute cycle. During a machine cycle one row of the matrix is activated. The first part of the row generates the control signals that control the operations of the processor. The second part generates the address of the row to be selected in the next machine cycle.

At the beginning of the cycle, the address of the row to be selected is contained in register I. This address is the input to the decoder, which is activated by a clock pulse.

This activates the row of the control matrix. The two-register arrangement is needed, as the decoder is a combinational circuit; with only one register, the output would become the input during a cycle. This may be an unstable condition due to repetitive loop.

4.5 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 general configuration of a micro-programmed control unit is demonstrated in Figure 5 below:

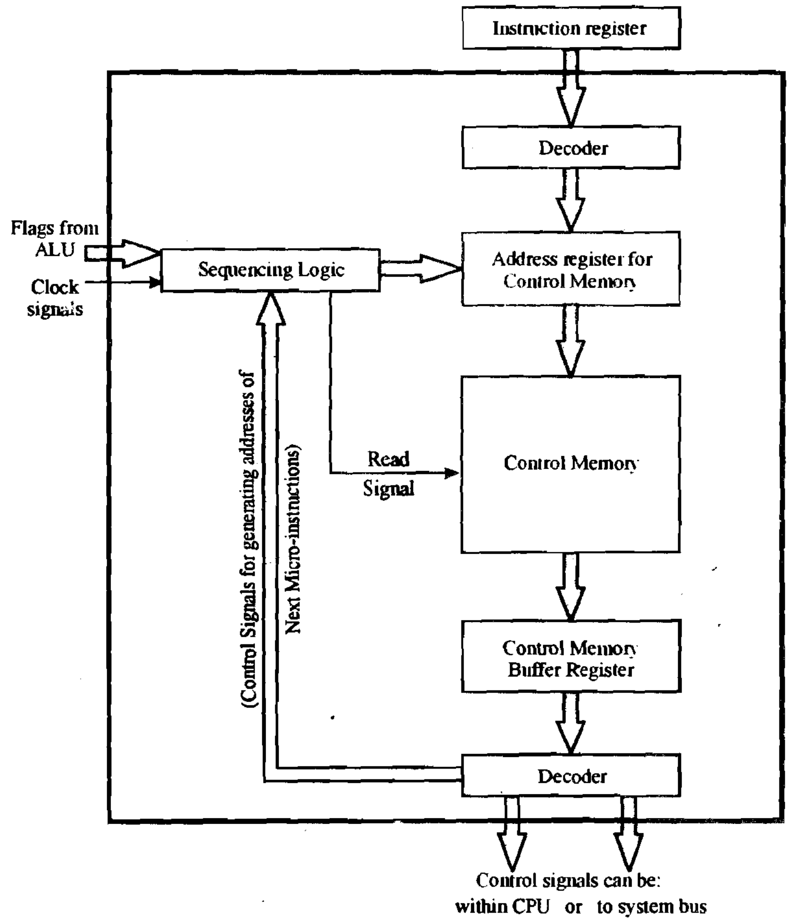


Figure 5: Operation of Micro-Programmed Control Unit

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:

1. 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.
2. The desired control memory word is read into control memory Buffer Register.
3. The content of the control memory buffer register is decoded to create control signals and next-address information for the sequencing logic unit.
4. 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.

As we have discussed earlier, 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 5). This decoder translates the opcode of the IR into a control memory address.

Check Your Progress 2

1. What is firmware? How is it different from software?

   ....

   ....

   ....

2. **State True or False** | T | F |

   (a) A micro-instruction can initiate only one micro-operation at a time. ☐

   (b) A control word is equal to a memory word. ☐

   (c) Micro-programmed control is faster than hardwired control. ☐

   (d) Wilkes control does not provide a branching micro-instruction. ☐

3. What will be the control signals and address of the next micro-instruction in the Wilkes control example of Figure 4, if the entry address for a machine instruction selects the last but one (branching control line) and the conditional bit value for branch is true?

   ....

   ....

   ....

4.6 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.

4.6.1 Types of Micro-instructions

In general the micro-instruction can be categorised into two general types. These are branching and 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.

4.6.2 Control Memory Organization

The next important question about the micro-instruction is: how are they organized in the control memory? 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 6 shows such an organisation.

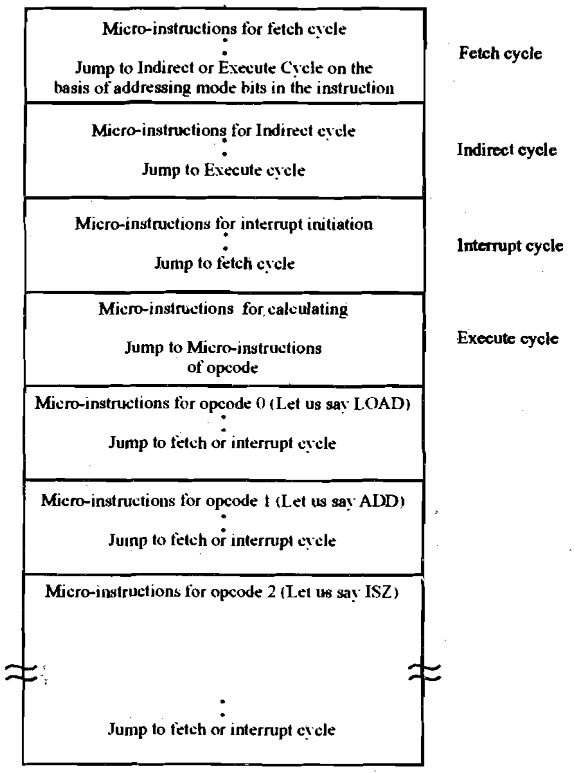

Micro-instructions for fetch cycle
Jump to Indirect or Execute Cycle on the basis of addressing mode bits in the instruction
Fetch cycle
Micro-instructions for Indirect cycle
Jump to Execute cycle
Indirect cycle
Micro-instructions for interrupt initiation
Jump to fetch cycle
Interrupt cycle
Micro-instructions for calculating
Jump to Micro-instructions of opcode
Execute cycle
Micro-instructions for opcode 0 (Let us say LOAD)
Jump to fetch or interrupt cycle
Micro-instructions for opcode 1 (Let us say ADD)
Jump to fetch or interrupt cycle
Micro-instructions for opcode 2 (Let us say ISZ)
Jump to fetch or interrupt cycle

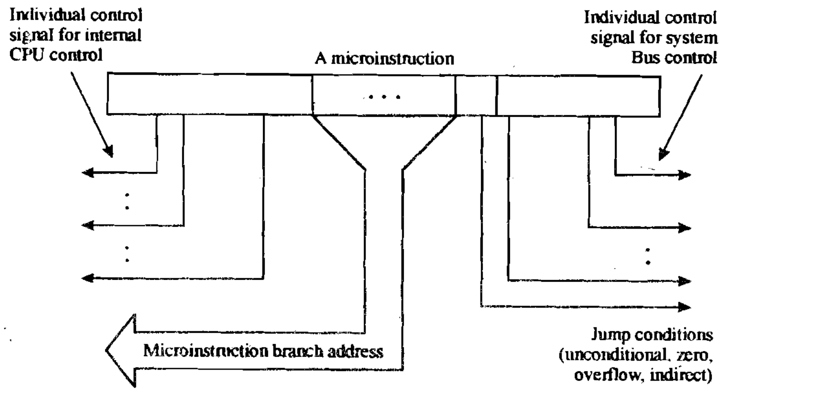
Figure 6: Control Memory Organisation

Let us give an example of control memory organization. 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 pseudocode of the micro-program for this instruction can be;

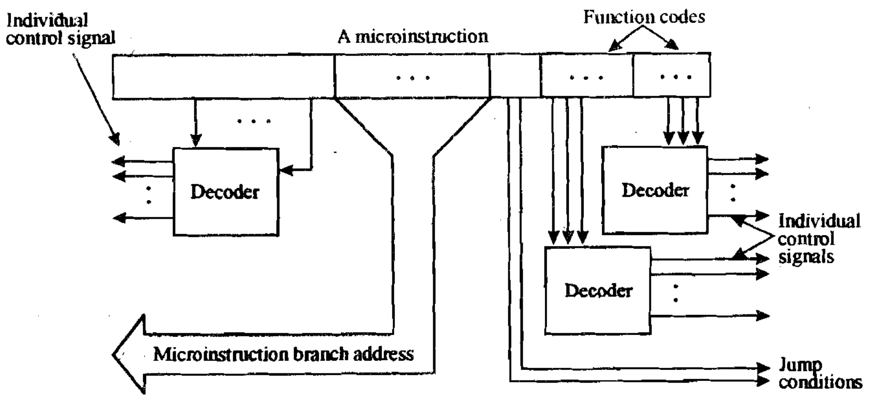
```
        Test "zero flag" If SET branch to label ZERO
        Unconditional branch to label NON-ZERO
ZERO: (Microcode which causes replacement of program counter with the address provided in the instruction)
    Branch to interrupt or fetch cycle.
NON -ZERO: (Microcode which may set flags if desired indicating the branch has not taken place).
    Branch to interrupt or fetch cycle. (For Next- Instruction Cycle)
```

4.6.3 Micro-instruction Formats

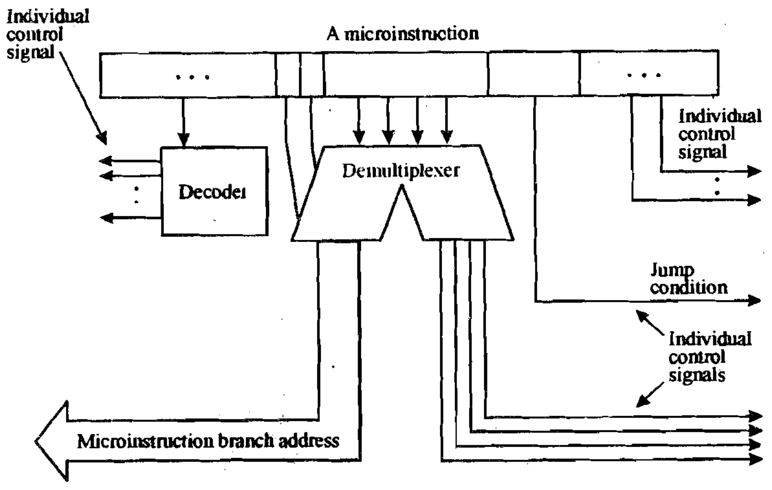
Now let us focus on the format of a micro-instruction. The two widely used formats used for micro-instructions are horizontal and vertical. In the horizontal micro-instruction each bit of the micro-instruction represents a control signal, which directly controls a single bus line or sometimes a gate in the machine. However, the length of such a micro-instruction may be hundreds of bits. A typical horizontal micro-instruction with its related fields is shown in Figure 7(a).


Individual control signal for internal CPU control
A microinstruction
Individual control signal for system Bus control
Microinstruction branch address
Jump conditions (unconditional, zero, overflow, indirect)

(a) Horizontal Micro-instruction


Individual control signal
A microinstruction
Function codes
Decoder
Decoder
Decoder
Individual control signals
Microinstruction branch address
Jump conditions

(b) Vertical Micro-instructions


Individual control signal
A microinstruction
Decoder
Demultiplexer
Individual control signal
Jump condition
Individual control signals
Microinstruction branch address

(c) A Realistic Micro-instructions
Figure 7: Micro- instruction Formats

In a vertical micro-instruction many similar control signals can be encoded into a few micro-instruction bits. For example, for 16 ALU operations, which may require 16 individual control bits in horizontal micro-instruction, only 4 encoded bits are needed in vertical micro-instruction. Similarly, in a vertical micro-instruction only 3 bits are needed to select one of the eight registers. However, these encoded bits need to be passed from the respective decoders to get the individual control signals. This is shown in figure 7(b).

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 7(c).

4.7 THE EXECUTION OF MICRO-PROGRAM

The micro-instruction cycle can consist of two basic cycles: the fetch and the execute. Here, in the fetch cycle the address of the micro-instruction is generated and this micro-instruction is put in a register used for the address of a micro-instruction for execution. The execution of a micro-instruction simply means generation of control signals. These control signals may drive the CPU (internal control signals) or the system bus. The format of micro-instruction and its contents determine the complexity of a logic module, which executes a micro-instruction.

One of the key features incorporated in a micro-instruction is the encoding of micro-instructions. What is encoding of micro-instruction? For answering this question let us recall the Wilkes control unit. In Wilkes control unit, each bit of information either generates a control signal or form a bit of next instruction address. Now, let us assume that a machine needs N total number of control signals. If we follow the Wilkes scheme we require N bits, one for each control signal in the control unit.

Since we are dealing with binary control signals, therefore, a 'N' bit micro-instruction can represent 2^N combinations of control signals.

The question is do we need all these 2^N combinations?

No, some of these 2^N combinations are not used because:

1. Two sources may be connected by respective control signals to a single destination; however, only one of these sources can be used at a time. Thus, the combinations where both these control signals are active for the same destination are redundant.
2. A register cannot act as a source and a destination at the same time. Thus, such a combination of control signals is redundant.
3. We can provide only one pattern of control signals at a time to ALU, making some of the combinations redundant.
4. We can provide only one pattern of control signals at a time to the external control bus also.

Therefore, we do not need 2^N combinations. Suppose, we only need 2^K (which is less than 2^N) combinations, then we need only K encoded bits instead of N control signals. The K bit micro-instruction is an extreme encoded micro-instruction. Let us touch upon the characteristics of the extreme encoded and unencoded micro-instructions:

Unencoded micro-instructions

- One bit is needed for each control signal; therefore, the number of bits required in a micro-instruction is high.
- It presents a detailed hardware view, as control signal need can be determined.

- Since each of the control signals can be controlled individually, therefore these micro-instructions are difficult to program. However, concurrency can be exploited easily.
- Almost no control logic is needed to decode the instruction as there is one to one mapping of control signals to a bit of micro-instruction. Thus, execution of micro-instruction and hence the micro-program is faster.
- The unencoded micro-instruction aims at optimising the performance of a machine.

Highly Encoded micro-instructions

- The encoded bits needed in micro-instructions are small.
- It provided an aggregated view that is a higher view of the CPU as only an encoded sequence can be used for micro-programming.
- The encoding helps in reduction in programming burden; however, the concurrency may not be exploited to the fullest.
- Complex control logic is needed, as decoding is a must. Thus, the execution of a micro-instruction can have propagation delay through gates. Therefore, the execution of micro-program takes a longer time than that of an unencoded micro-instruction.
- The highly encoded micro-instructions are aimed at optimizing programming effort.

In most of the cases, the design is kept between the two extremes. The LSI 11 (highly encoded) and IBM 3033 (unencoded) control units are close examples of these two approaches.

Execution/decoding of slightly encoded micro-instructions

In general, the micro-programmed control unit designs are neither completely unencoded nor highly encoded. They are slightly coded. This reduces the width of control memory and micro-programming efforts. The basic technique for encoding is shown in Figure 8. The micro-instruction is organised as a set of fields. Each field contains a code, which, upon decoding, activates one or more control signals. The execution of a micro-instruction means that every field is decoded and generates control signals. Thus, with N fields, N simultaneous actions can be specified. Each action results in the activation of one or more control signals. Generally each control signal is activated by no more than one field. The design of an encoded micro-instruction format can be stated in simple terms:

- Organize the format into independent fields. That is, each field depicts a set of actions such that actions from different fields can occur simultaneously.
- Define each field such that the alternative actions that can be specified by the field are mutually exclusive. That is, only one of the actions specified for a given field could occur at a time.

Another aspect of encoding is whether it is direct or indirect (Figure 8). With indirect encoding, one field is used to determine the interpretation of another field.

Another aspect of micro-instruction execution is the micro-instruction sequencing that involves address calculation of the next micro-instruction. In general, the next micro- instruction can be (refer Figure 6):

- Next micro-instruction in sequence
- Calculated on the basis of opcode
- Branch address (conditional or unconditional).

A detailed discussion on these topics is beyond this unit. You must refer to further readings for more detailed information on Micro-programmed Control Unit Design.

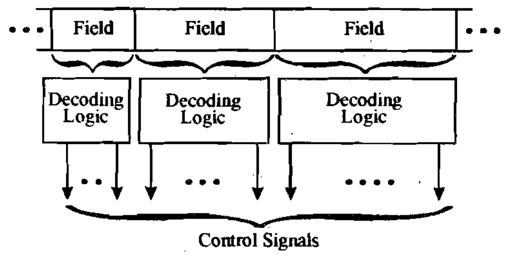


Figure (a): Direct Encoding

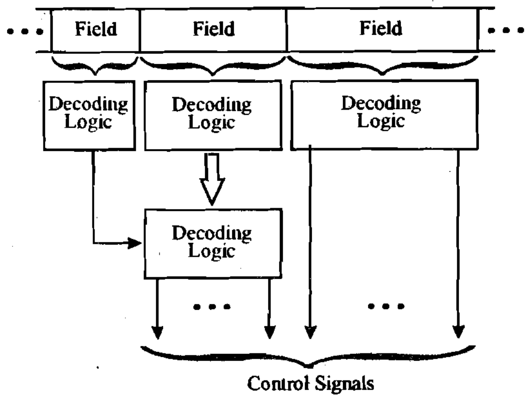


Figure (b): Indirect Encoding

Figure 8: Micro-instruction Encoding

Check Your Progress 3

T	F

1. **State True or False**

   a) A branch micro-instruction can have only an unconditional jump. ☐

   b) Control store stores opcode-based micro-programs. ☐

   c) A true horizontal micro-instruction requires one bit for every control signal. ☐

   d) A decoder is needed to find a branch address in the vertical micro-instruction. ☐

   e) One of the responsibilities of sequencing logic (Refer Figure 5) is to cause reading of micro-instruction addressed by a micro-program counter into the micro-instruction buffer. ☐

   f) Status bits supplied from ALU to sequencing logic have no role to play with the sequencing of micro-instruction. ☐

2. What art the possibilities for the next instruction address?

   ....

   ....

   ....

   ....

   ....

3. How many address fields are there in Wilkes Control Unit?

   ....

   ....

   ....

4. Compare and contrast unencoded and highly encoded micro-instructions.

   ....

   ....

   ....

4.8 SUMMARY

In this unit we have discussed the organization of control units. Hardwired, Wilkes and micro-programmed control units are also discussed. The key to such control units are micro-instruction, which can be briefly (that is types and formats) described in this unit. The function of a micro-programmed unit, that is, micro-programmed execution, has also been discussed. The control unit is the key for the optimised performance of a computer. The information given in this unit can be further appended by going through further readings.

4.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. IR, Timing Signal, Flags Register
2. The control unit issues control signals that cause execution of micro-operations in a pre-determined sequence. This, enables execution sequence of an instruction.
3. A logic circuit based implementation of control unit.

Check Your Progress 2

1. Firmware is basically micro-programs, which are used in a micro-programmed control unit. Firmwares are more difficult to write than software.

2. (a) False (b) False (C) False (d) False

3. In sequence from left to right as per figure.
   110......00 (control signals ...... indicate more values)
   110......00 (address of next, micro-instruction is found after assuming that bottom line after condition code represent true in the Figure 4)

Check Your Progress 3

1. (a) False (b) False (c) True (d) False (e) True (f) False.

2 The address of the next micro-instruction can be one of the following:

   - the address of the next micro-instruction in sequence.
   - determined by opcode using mapping or any other method.
   - branch address supplied on the internal address bus.

3. Wilkes control typically has one address field. However, for a conditional branching micro-instruction, it contains two addresses. The Wilkes control, in fact, is a hardware representation of a micro-programmed control unit.

4.

Unencoded Micro instructions	Highly encoded
• Large number of bits	Relatively less bits
• Difficult to program	Easy to program
• No decoding logic	Need decoding logic
• Optimizes machine performances	Optimizes programming effort
• Detailed hardware view	Aggregated view