UNIT-5

❖ MICROPROGRAMMED CONTROL UNIT

The term *microprogram* was first coined by M. V. Wilkes in the early 1950s. Wilkes proposed an approach to control unit design that was organized and systematic that avoided the complexities of a hardwired implementation. An alternative for Hardwired Control Unit, which has been used in many CISC processors, is to implement a microprogrammed control unit.

A language is used to deign microprogrammed control unit known as a **microprogramming language**. Each line describes a set of micro-operations occurring at one time and is known as a **microinstruction**. A sequence of instructions is known as a **microprogram**,or *firmware.*

For each micro-operation, control unit has to do is generate a set of control signals. Thus, for any micro-operation, control line emanating from the control unit is either on or off. This condition can, be represented by a binary digit for each control line. So we could construct a *control word* in which each bit represents one control line. Now add an address field to each control word, indicating the location of the next control word to be executed if a certain condition is true.

The result is known as a **horizontal microinstruction**, which is shown in Figure 5.1a. The format of the microinstruction or control word is as follows. There is one bit for each internal processor control line and one bit for each system bus control line. There is a condition field indicating the condition under which there should be a branch, and there is a field with the address of the microinstruction to be executed next when a branch is taken. Such a microinstruction is interpreted as follows:

1 To execute this microinstruction, turn on all the control lines indicated by a 1 bit;leave off all control lines indicated by a 0 bit.The resulting control signals will cause one or more micro-operations to be performed.

2 If the condition indicated by the condition bits is false, execute the next microinstruction in sequence.

3 If the condition indicated by the condition bits is true, the next microinstruction to be executed is indicated in the address field.

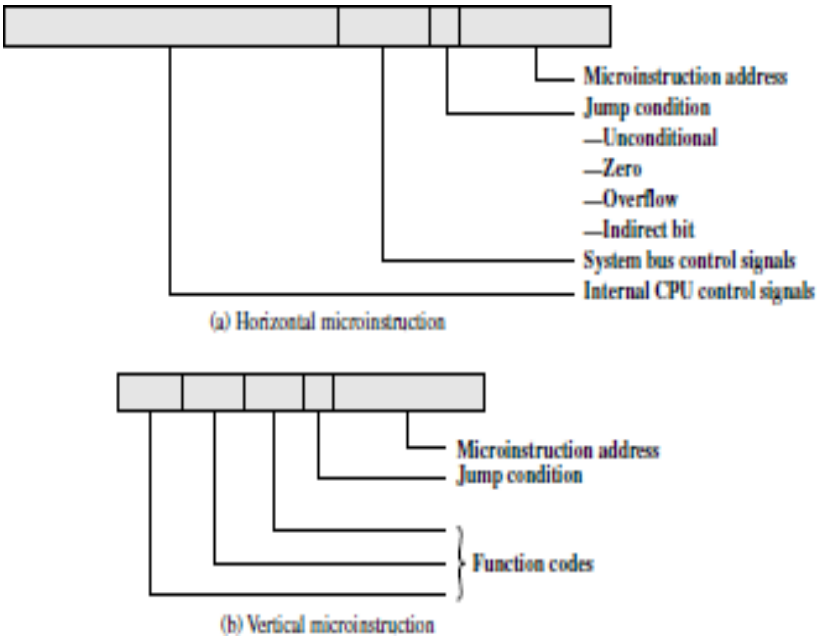

Figure 5.1 Typical MicroInstruction Format

In a **vertical microinstruction**, a code is used for each action to be performed, and the decoder translates this code into individual control signals. The advantage of vertical microinstructions is that they are more compact (fewer bits) than horizontal microinstructions.

Figure 5.2 shows how these control words or microinstructions could be arranged in a **control memory**. The microinstructions in each routine are to be executed sequentially. Each routine ends with a branch or jump

instruction indicating where to go next. There is a special execute cycle routine whose only purpose is to signify that one of the machine instruction routines (AND,ADD,and so on) is to be executed next, depending on the current opcode.

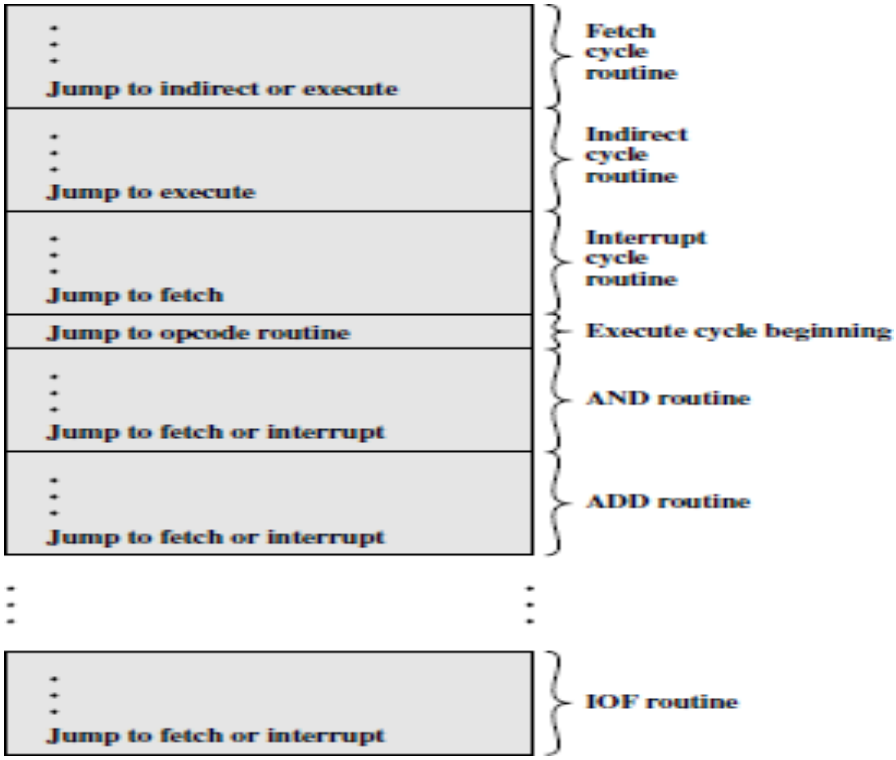

Figure 5.2 Organization of Control Memory

The Figure 5.2 defines the sequence of micro-operations to be performed during each cycle (fetch, indirect, execute, interrupt), and it specifies the sequencing of these cycles.

Microprogrammed Control Unit

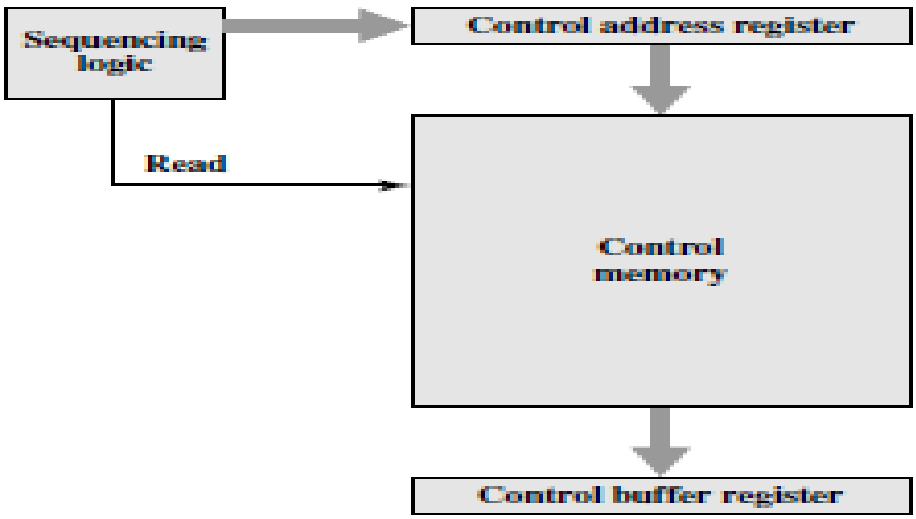

Figure 5.3 Control Unit MicroArchitecture

Figure 5.3 shows the key elements of a microprogrammed control unit implementation. The set of microinstructions is stored in the *control memory*. The *control address register* contains the address of the next microinstruction to be read. When a microinstruction is read from the control memory, it is transferred to a *control buffer register*. The left-hand portion of that register (see Figure 5.1a) connects to the control lines emanating from

the control unit. Thus, *reading* a microinstruction from the control memory is the same as *executing* that microinstruction.The third element shown in the figure is a sequencing unit that loads the control address register and issues a read command.

Let us examine this structure in greater detail, as depicted in Figure 5.4.The control unit functions as follows:

1 To execute an instruction, the sequencing logic unit issues a READ command to the control memory.

2 The word whose address is specified in the control address register is read into the control buffer register.

3 The content of the control buffer register generates control signals and next-address information for the sequencing logic unit.

4 The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffer register and the ALU flags.

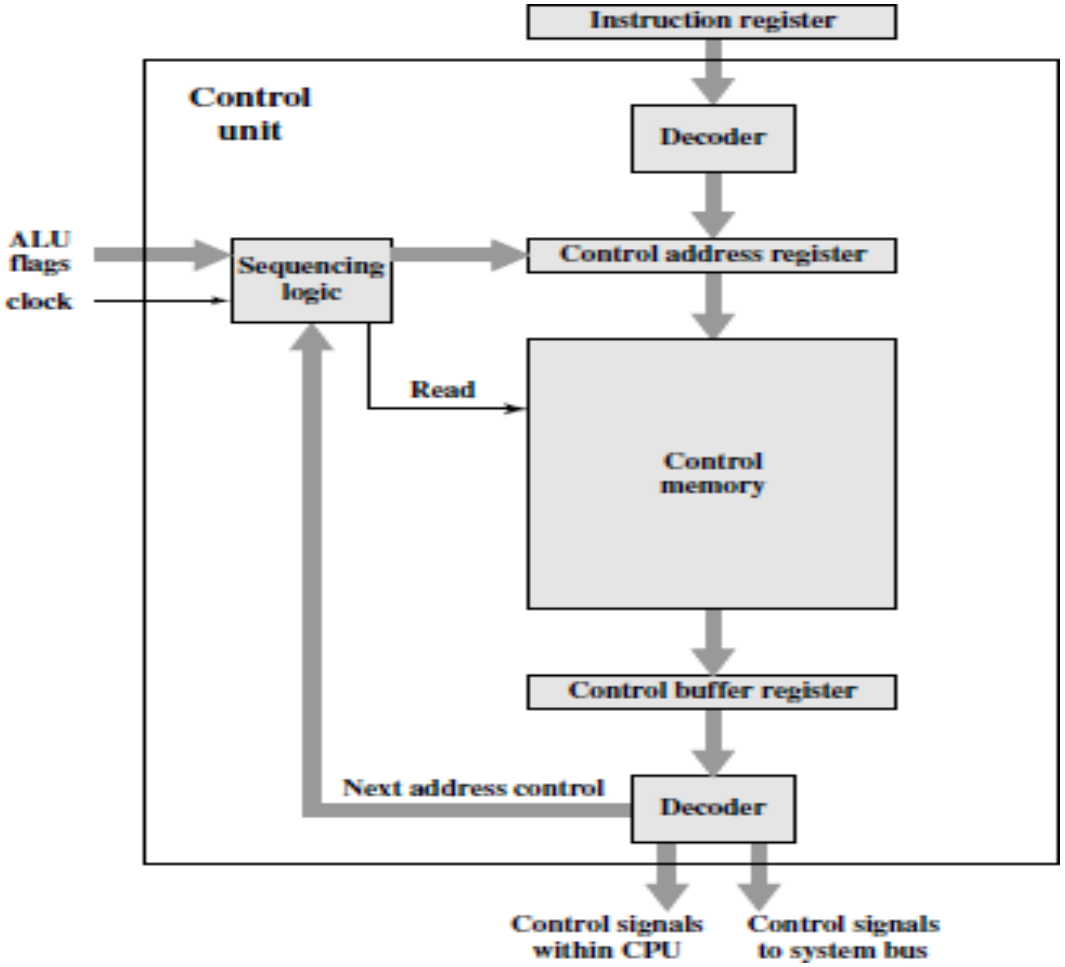

Figure 5.4 Functioning of microprogrammed control unit

At the conclusion of each microinstruction, the sequencing logic unit loads a new address into the control address register. Depending on the value of the ALU flags and the control buffer register, one of three decisions is made:

- **Get the next instruction:** Add 1 to the control address register.
- **Jump to a new routine based on a jump microinstruction:** Load the address field of the control buffer register into the control address register.
- **Jump to a machine instruction routine:** Load the control address register based on the opcode in the IR.

Figure 5.4 shows two modules labeled *decoder*. The upper decoder translates the opcode of the IR into a control memory address. The lower decoder is not used for horizontal microinstructions but is used for **vertical microinstructions** (Figure 5.1b). In a horizontal microinstruction every bit in the control field attaches to a control line. In a vertical microinstruction, a code is used for each action to be performed, and the decoder translates this code into individual control signals. The advantage of vertical microinstructions is that they are more compact (fewer bits) than horizontal microinstructions.

Advantages and Disadvantages

The principal advantage of the use of microprogramming to implement a control unit is that it simplifies the design of the control unit. Thus, it is both cheaper and less error prone to implement. A *hardwired* control unit must contain complex logic for sequencing through the many micro-operations of the instruction cycle. On the other hand, the decoders and sequencing logic unit of a microprogrammed control unit are very simple pieces of logic.

The principal disadvantage of a microprogrammed unit is that it will be somewhat slower than a hardwired unit of comparable technology. Despite this, microprogramming is the dominant technique for implementing control units in pure CISC architectures, due to its ease of implementation. RISC processors, with their simpler instruction format, typically use hardwired control units.

❖ MICROINSTRUCTION SEQUENCING

The two basic tasks performed by a microprogrammed control unit are as follows:

- **Microinstruction sequencing:** Get the next microinstruction from the control memory.
- **Microinstruction execution:** Generate the control signals needed to execute the microinstruction.

In designing a control unit, these tasks must be considered together, because both affect the format of the microinstruction and the timing of the control unit.

Design Considerations

Two concerns are involved in the design of a microinstruction sequencing technique: the size of the microinstruction and the address-generation time. The first concern is obvious; minimizing the size of the control memory reduces the cost of that component. The second concern is simply a desire to execute microinstructions as fast as possible.

In executing a microprogram, the address of the next microinstruction to be executed is in one of these categories:

- Determined by instruction register
- Next sequential address
- Branch

Sequencing Techniques

Based on the current microinstruction, condition flags, and the contents of the instruction register, a control memory address must be generated for the next microinstruction. A wide variety of techniques have been used. These categories are based on the format of the address information in the microinstruction:

- Two address fields
- Single address field
- Variable format

Two Address Field:

The simplest approach is to provide two address fields in each microinstruction. Figure 5.5 suggests how this information is to be used. A multiplexer is provided that serves as a destination for both address fields plus the instruction register. Based on an address-selection input, the multiplexer transmits either the opcode or one of the two addresses to the control address register (CAR). The CAR is subsequently decoded to produce the next microinstruction address.The address-selection signals are provided by a branch logic module whose input consists of control unit flags plus bits from the control portion of the microinstruction.

Although the two-address approach is simple, it requires more bits in the microinstruction than other approaches. With some additional logic, savings can be achieved.

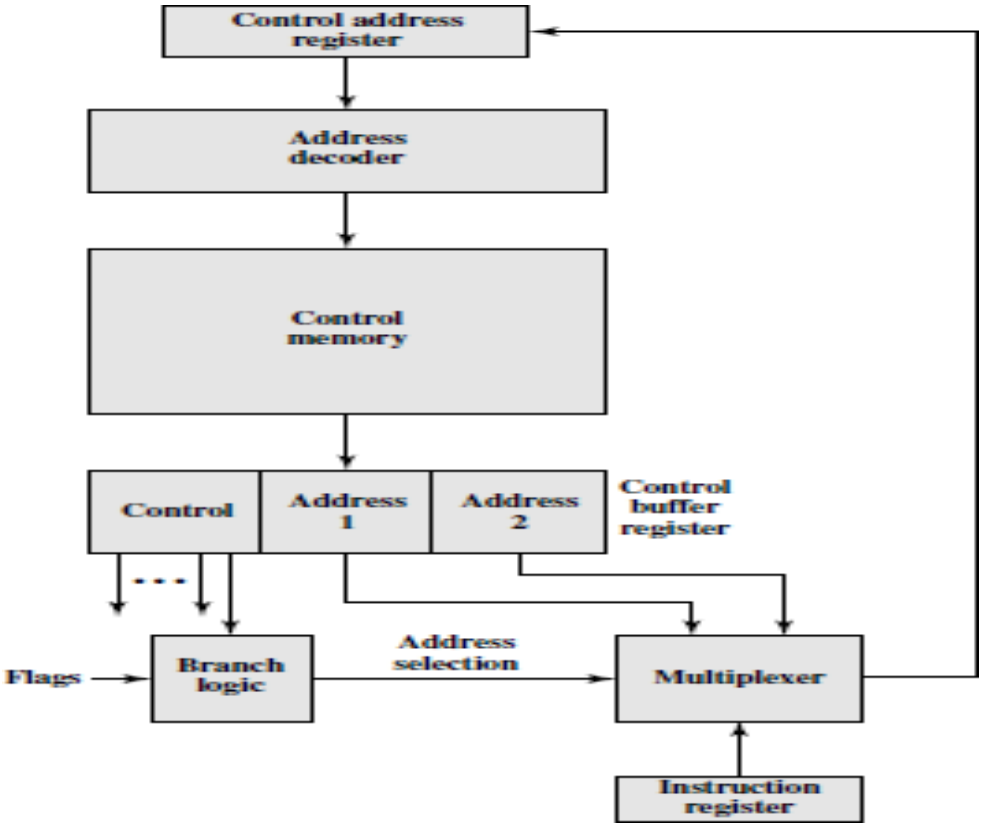

Figure 5.5 Branch Control Logic: Two Address Field

Single Address Field:

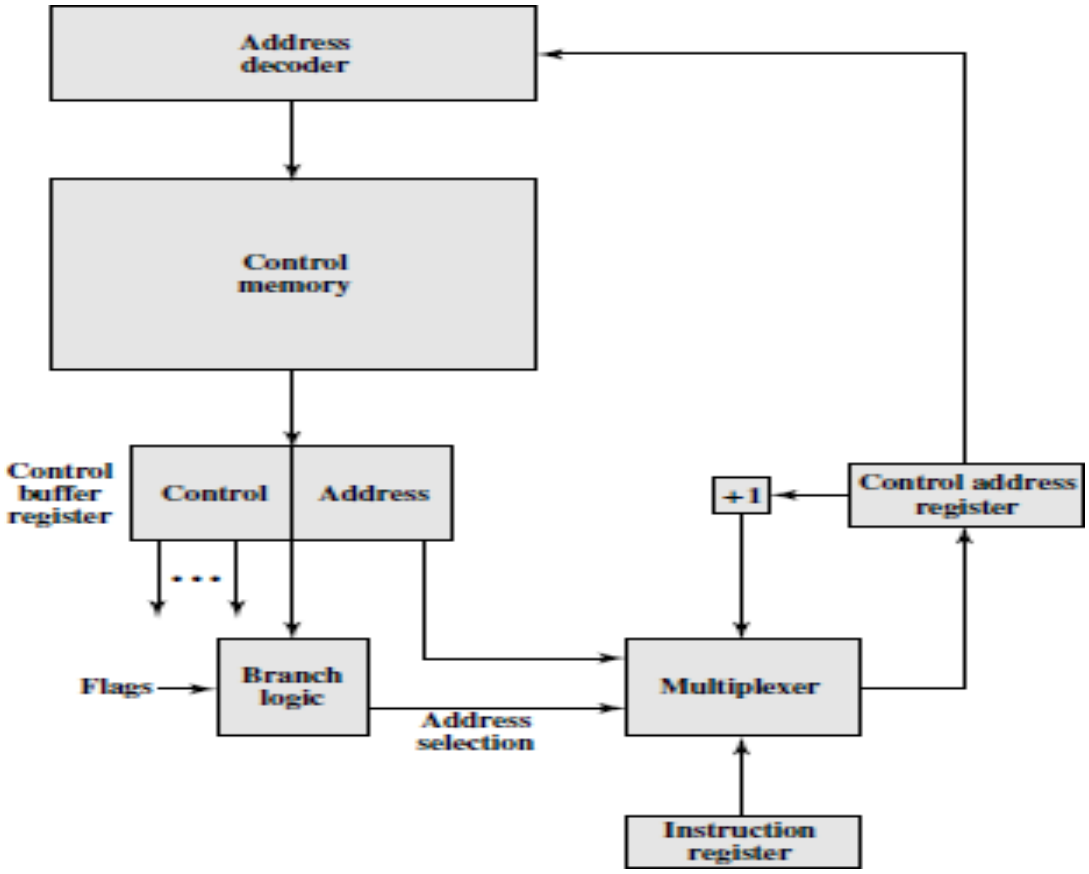

Figure 5.6 Branch Control Logic: Single Address Field

A common approach is to have a single address field (Figure 5.6). With this approach, the options for next address are as follows:

- Address field
- Instruction register code
- Next sequential address

The address-selection signals determine which option is selected. This approach reduces the number of address fields to one. Note, however, that the address field often will not be used. Thus, there is some inefficiency in the microinstruction coding scheme.

Variable Format:

Another approach is to provide for two entirely different microinstruction formats (Figure 5.7). One bit designates which format is being used. In one format, the remaining bits are used to activate control signals. In the other format, some bits drive the branch logic module, and the remaining bits provide the address.

One disadvantage of this second approach is that one entire cycle is consumed with each branch microinstruction. With the other approaches, address generation occurs as part of the same cycle as control signal generation, minimizing control memory accesses.

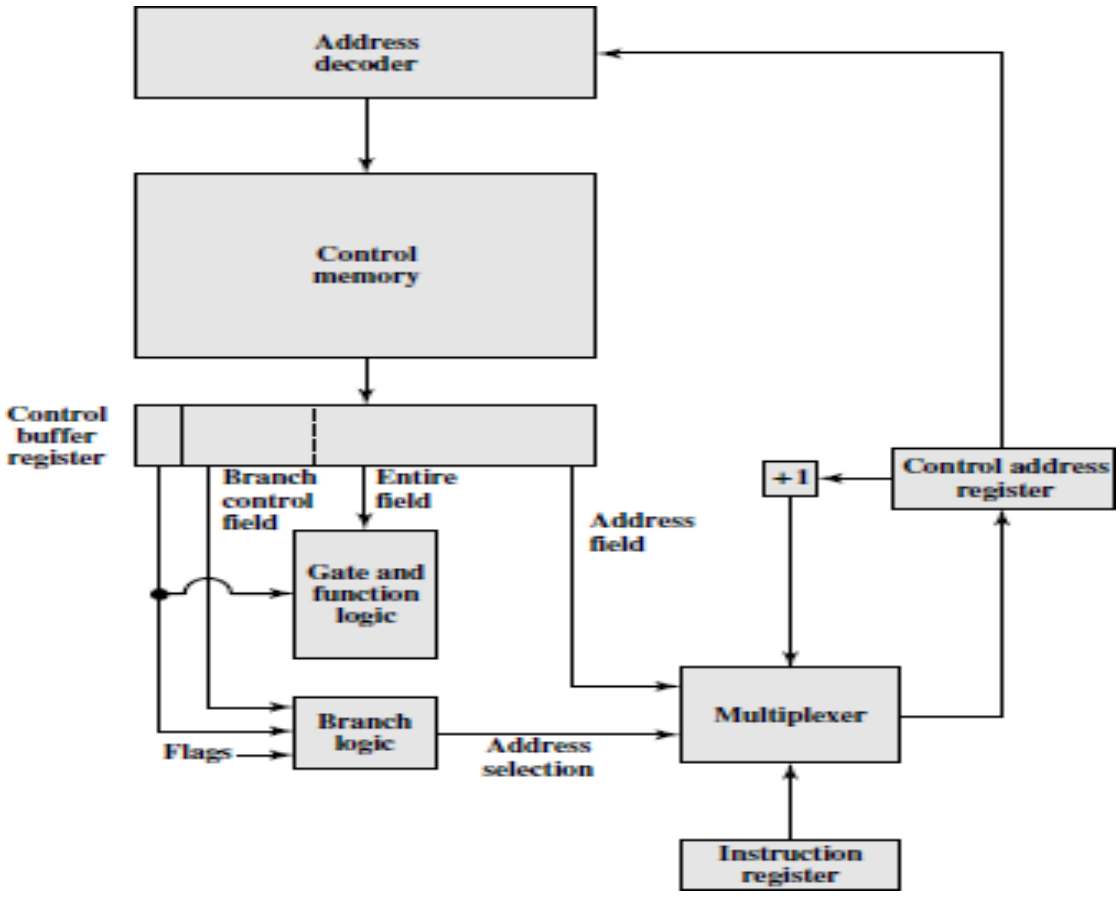

Figure 5.7 Branch Control Logic: Variable Format

Address Generation

There are various ways in which the next address can be derived or computed.

Table 5.1 lists the various address generation techniques. These can be divided into explicit techniques, in which the address is explicitly available in the microinstruction, and implicit techniques, which require additional logic to generate the address.

Explicit	Implicit
Two-field	Mapping
Unconditional branch	Addition
Conditional branch	Residual control

Table 5.1 MicroInstruction Address Generation Techniques

We have essentially dealt with the explicit techniques. With a two-field approach, two alternative addresses are available with each microinstruction. Using either a single address field or a variable format, various branch instructions can be implemented. A conditional branch instruction depends on the following types of information:

- ALU flags
- Part of the opcode or address mode fields of the machine instruction
- Parts of a selected register, such as the sign bit
- Status bits within the control unit

Several implicit techniques are also commonly used. One of these, mapping, is required with virtually all designs. The opcode portion of a machine instruction must be mapped into a microinstruction address.This occurs only once per instruction cycle. A common implicit technique is Addition that involves combining or adding two portions of an address to form the complete address. The final approach is termed *residual control*. This approach involves the use of a microinstruction address that has previously been saved in temporary storage within the control unit. An example of this approach is taken on the LSI-11.

LSI-11 Microinstruction Sequencing

The LSI-11 is a microcomputer version of a PDP-11, which is implemented using a microprogrammed control unit.

The LSI-11 makes use of a 22-bit microinstruction and a control memory of 2K 22-bit words. The next microinstruction address is determined in one of five ways:

- **Next sequential address:** In the absence of other instructions, the control unit's control address register is incremented by 1.
- **Opcode mapping:** At the beginning of each instruction cycle, the next microinstruction address is determined by the opcode.
- **Subroutine facility:** Explained presently.
- **Interrupt testing:** Certain microinstructions specify a test for interrupts. If an interrupt has occurred, this determines the next microinstruction address.
- **Branch:** Conditional and unconditional branch microinstructions are used.

❖ MICROINSTRUCTION EXECUTION

The microinstruction cycle is the basic event on a microprogrammed processor. Each cycle is made up of two parts: fetch and execute. The fetch portion is determined by the generation of a microinstruction address and the execution of a microinstruction is to generate control signals. Some of these signals control points internal to the processor. The remaining signals go to the external control bus or other external interface.

A Taxonomy of Microinstructions

Microinstructions can be classified in a variety of ways. Distinctions that are commonly made in the literature include the following:

- Vertical/horizontal
- Packed/unpacked
- Hard/soft microprogramming
- Direct/indirect encoding

Horizontal/Vertical microinstruction: Each bit of a microinstruction either directly produced a control signal or directly produced one bit of the next address. These schemes require a more complex sequencing logic module. **Packed/Unpacked Microinstruction**: The degree of packing relates to the degree of identification between a given control task and specific microinstruction bits. As the bits become more *packed*, a given number of bits contains more information. Thus, packing connotes encoding. The terms *horizontal* and *vertical* relate to the relative width of microinstructions. The terms ***hard* and *soft* microprogramming** are used to suggest the degree of closeness to the underlying control signals and hardware layout. Hard microprograms are generally fixed and committed to read-only memory. Soft microprograms are more changeable and are suggestive of user microprogramming. The other pair is **direct versus indirect encoding**.

Microinstruction Encoding

Some degree of encoding is used to reduce control memory width and to simplify the task of microprogramming.

The basic technique for encoding is illustrated in Figure 5.8a. The microinstruction is organized as a set of fields. Each field contains a code, which, upon decoding, activates one or more control signals. When the microinstruction is executed, every field is decoded and generates control signals.Thus, with N fields, N simultaneous actions are specified. Each action results in the activation of one or more control signals

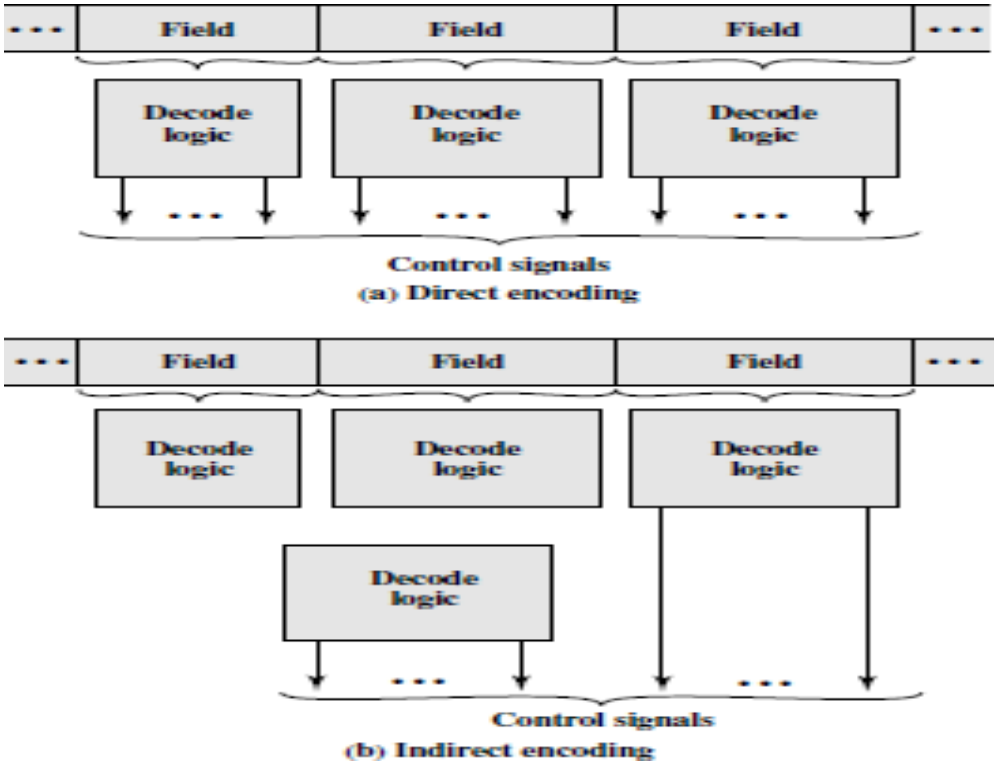

Figure 5.8 MicroInstruction Encoding

The design of an encoded microinstruction format can now be stated in simple terms:

- Organize the format into independent fields. That is, each field depicts a set of actions (pattern of control signals) 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.

Two approaches can be taken to organizing the encoded microinstruction into fields: functional and resource. The ***functional encoding*** method identifies functions within the machine and designates fields by function type. For example, if various sources can be used for transferring data to the accumulator, one field can be designated for this purpose, with each code specifying a different source. ***Resource encoding*** views the machine as consisting of a set of independent resources and devotes one field to each (e.g., I/O, memory, ALU).

Another aspect of encoding is whether it is direct or indirect (Figure 5.8b). With indirect encoding, one field is used to determine the interpretation of another field.