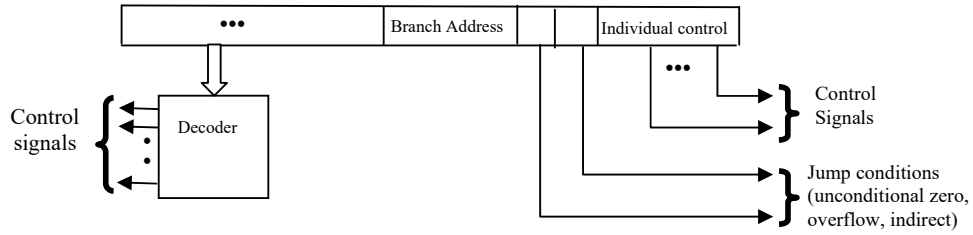

(c) A Realistic Micro-instruction

Figure 11.6: 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 11.6(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 11.6(c).

11.7 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 processor (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.These logic module depends on the encoding of micro-instructions, which is discussed next.

11.7.1 Micro-instruction Encoding

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 youare using the Wilkes scheme you require N bits for the control signals, one for each control signal in the control unit. In addition, Wilkes control unit also stores the address of the next micro-instruction using address bits.

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. You can provide only one pattern of control signals at a time to ALU, making some of the combinations redundant.
4. You can provide only one pattern of control signals at a time to the external control bus also.

Therefore, you do not need 2^N combinations. Supposeyou only need 2^K (where K< N) combinations, then you 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 smaller in size than that ofunencoded micro-instructions.
- It provided an aggregated view that is a higher view of the processor 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 longer time than that of an unencoded micro-instruction.
- The highly encoded micro-instructions are aimed at optimizing programming effort.

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. As shown in Figure 11.7, some of the bits of micro-instructions are unencoded, therefore, can be used to generate the control signals directly. Some of the control bits are organised as fields and can be directly used as input to decoder to generate control signals. Further, a combination of decoded control signals can be passed through a second decoder to generate control signals. These decoding operations are shown in Figure 11.7.

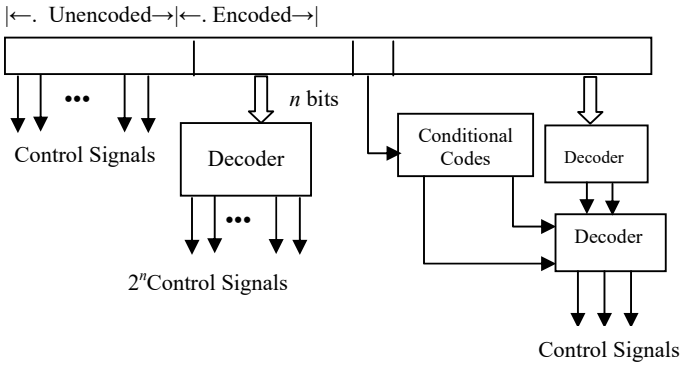

Figure 11.7: Decoding of Micro-instructions

11.7.2 Micro-instruction Sequencing

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 one of the following (refer Figure 11.5):

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

Next instruction in sequence: Figure 11.5 shows one example of control memory. This control organisation has micro-instructions for fetch, indirect and interrupt cycles followed by the execute cycle. You may recall the instruction cycle, as given in Unit 10, the fetchinstruction cycle consists of the following sequence of micro-operations:

T1: MAR ← PC
T2: DR ← (MAR)
T3: PC←PC +1; IR ← DR

One micro-instruction can be created for each timing sequence. For example, for the stated sequence of micro-operations, three micro-instructions, one each for timing T1, T2 and T3 would be created. These micro-instructions would be stored at address 00h, 01h and 02h. The micro-instruction at 03h would be a conditional branch instruction to the indirect or execute cycle. Please also note that at time T3, two micro-operations are to be performed in parallel. Thus, micro-instruction at 02h should generate control signals, which result in the related units of the processor to perform the increment operation on PC and transfer of DR to IR simultaneously.
The micro-instruction at 00h to 03h are to be executed in a sequence to perform the desired operation of instruction fetch.

Branch address (conditional or unconditional): You may please note that the last micro-instruction for instruction fetch is the conditional branch instruction. Please also note that in Figure 11.5, the indirect cycle starts at an address 04h and the execution cycle starts at 0Ch. Thus, the micro-instruction at address 03h would be a conditional branch instruction, having the condition, if the indirect bit is set of not. This conditional branch would be taken to address 0Ch (the start of execution cycle) in case the indirect bit is CLEAR.Thus, if indirect bit is SET, then the next micro-instruction in sequence would be executed, which will be the starting micro-instruction of the indirect cycle that will convert the indirect operand to direct operand. Please also note in the Figure 11.5, that the last micro-instruction of the indirect cycle is an unconditional jump instruction to the execute cycle.

Calculated on the basis of opcode: The opcode of the Instruction Register is used to decode the operation that is to be performed on the operands. The control unit supports this decode operation. In the case of micro-programmed control unit, this opcode can be used to compute the address of the first micro-instruction to be executed to perform the operation. In Figure 11.5, the execute cycle contains the

micro-instructions that perform jump to the micro-instruction address of the desired operation.
We will explain it with the help of an example, assume that in Figure 11.5, the micro-instructions related to opcodes start from micro-instruction address 00h instead of 10h and the micro-instructions of fetch, indirect, interrupt and execute cycles starts at micro-instruction address F0h, F4h, F8h and FCh respectively.Further, we assume that operation of each opcode is performed using just four micro-instructions. Since the control memory has addresses from 00h to FFh, out of which 00h to EFh(a total of F0h) addresses are for storing micro-instructions of various opcodes. Therefore, this control memory can contain micro-instructions for F0h/4h = 3Ch opcodes. Thus, the possible opcodes for such a machine would be $00000000_2$to 00111011_2, or 000000_2 to 111011_2.How these opcodes would be mapped to the related micro-instruction start address. The following table shows these mapping:

Opcode	Starting address of related Micro-instructions	
Binary	Binary	Hexadecimal
0000 00	0000 0000	00
0000 01	0000 0100	04
0000 10	0000 1000	08
0000 11	0000 1000	0C
0001 00	0001 0000	10
0001 01	0001 0100	14
0001 10	0001 1000	18
0001 11	0001 1000	1C
…	…	
1001 00	1001 0000	90
1001 00	1001 0100	94
…	…	
1110 00	1110 0000	E0
1110 01	1110 0100	E4
1110 10	1110 1000	E8
1110 11	1110 1000	EC

Figure 11.8: Mapping of opcode to micro-instruction address

The interesting part of this example is the mapping from the opcode to the micro-instruction address. In Figure 11.8, an opcode is of 6-bit length. To map an opcode to the first related micro-instruction, you just need to append two 0 bits at the least significant position.For example, an opcode 1001 01 will be mapped to a micro-instruction address 1001 0100 or 94h.

Please note different kind of control memory and opcode organisation will make this computational logic a complex one. In any case, you can design the related logic circuit for calculating the micro-instruction address for a given opcode.

You must refer to further readings for more detailed information on Micro-programmed Control Unit Design.

☞Check Your Progress 3

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 11.4) is to cause reading of micro-instruction addressed by a micro-program address register. ☐

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.

.....

.....

.....

11.8 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.

11.9 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. Firmwareare more difficult to write than software.

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

3. Please check the Figure 11.3 from left to right and select the bottom branch line.
   The control signals would be 000…00
   Address of next micro-instruction would be: 100…10

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 • Difficult to program • No decoding logic • Optimizes machine performances • Detailed hardware view	Relatively less bits Easy to program Need decoding logic Optimizes programming effort Aggregated view