Chapter-3, Control Unit

Micro-operations:-

Micro by its name gives that operation are small and simple. Micro-operations are smaller series of steps each of which involves the processor registers.

Instruction cycle is made up of a number of smaller units. One subdivision is fetch ,indirect, execute and interrupt. Execution of a program consists of sequential execution of instructions. Instruction is executed during an instruction cycle made up of shorter sub cycle. The performance of each sub cycle involvesmicro operations. These are the functional or atomic operations of a processor.

Sub cycle of an instruction cycle

1. **The fetch cycle**

   It is the beginning of each instruction cycle which fetch an instruction from memory. Four registers are involved:

   Memory Address Register (MAR)
   Memory Buffer Register (MBR)
   Program Counter (PC)
   Instruction Register (IR)

   Memory Address register specifies address in memory for read or write operation. It is connected with address line. Memory Buffer register is connected to data lines to store the value in memory or the last value from memory. PC holds address of next instruction to be fetched. IR holds last instruction fetched.

   At the beginning the address of the next instruction to be executed is in Program Counter (PC). The first step is this address is moved to MAR. The second step is to bring the instruction. MAR place the address in address bus and issues READ command. The result appears in data bus and is copied in MBR. Pc should be incremented by 1 for next instruction. These two instruction do not interfere each other thus can be done simultaneously. The third step is to move the content of MBR to IR.

   Thus a simple fetch cycle consists of three steps and four micro operations:

   t1: MAR ← (PC)
   t2: MBR ←Memory,PC← (PC)+1

   t3: IR ← (MBR)

   the notation (t1,t2,t3) represents successive time units.

   The grouping of micro operations must follow two simple rules:

   1. The proper sequence of events must be followed. Thus MAR ← (PC) must precede MBR ← Memory because the memory read operation makes use of address in MAR.
   2. Conflicts must be avoided. Read and write should not be done from same register in one time unit.

2. The indirect cycle

Once an instruction is fetched, the next step is tol fetch source operands. Let us assume one address instruction format with direct and indirect addressing. If instruction specifies indirect address, then indirect cycle must precede execute cycle. This includes following micro operations:

t1: MAR ← (IR (Address))
t2: MBR ← Memory
t3: IR(Address) ← (MBR(Address))

Address field of instruction is transferred to MAR. This is used to fetch the address o operand. Address of IR is updated from MBR, so that it contains direct address rather than indirect. Now it is ready to execute.

3. The interrupt cycle

After execution cycle, test is performed for existence of any interrupts. If there is interrupt, interrupt cycle occurs.

t1: MBR ← (PC)
t2: MAR ←Save_Address, PC ←Routine_Address
t3: Memory ← (MBR)

In first step content of PC are transferred to MBR, so that they can be saved for return from interrupt. MAR is loaded with address at which PC contents are to be saved and PC with address of interrupt. In third step MBR, content is store in memory. The processor then begins with next instruction cycle. It varies from machine to machine.

4. The execute cycle

For execute cycle it is difficult to predict the micro-operations. For a machine with N different opcodes there are N different sequences of micro-operations. Eg for add operation.

— **ADD R1, X**
t1: MAR ← (IR (Address))
t2: MBR ←Memory
t3: R1← (MBR)+(R1)
— **ISZ X (increment and skip if zero)**
t1: MAR ← (IR (Address))
t2: MBR ←Memory
t3: MBR ← (MBR) + 1
t4: Memory ← (MBR)
if ((MBR) = 0) then (PC ← (PC) + I)

The Instruction Cycle

Fetch, Indirect and Interrupt cycle has one sequence of operations but for execute cycle there is one sequence of micro- operations for each opcodes. All these micro-operations are bring together to complete an instruction cycle. For this we have one two-bit register called Instruction Cycle Code(ICC).

00: Fetch
01: Indiect
10: Execute

11: Interrupt

At the end of four cycles, the ICC is set appropriately and interrupt cycle is followed by fetch cycle. For both execute and fetch cycle the next cycle depends on the state of system.

Flow chart for Instruction cycle

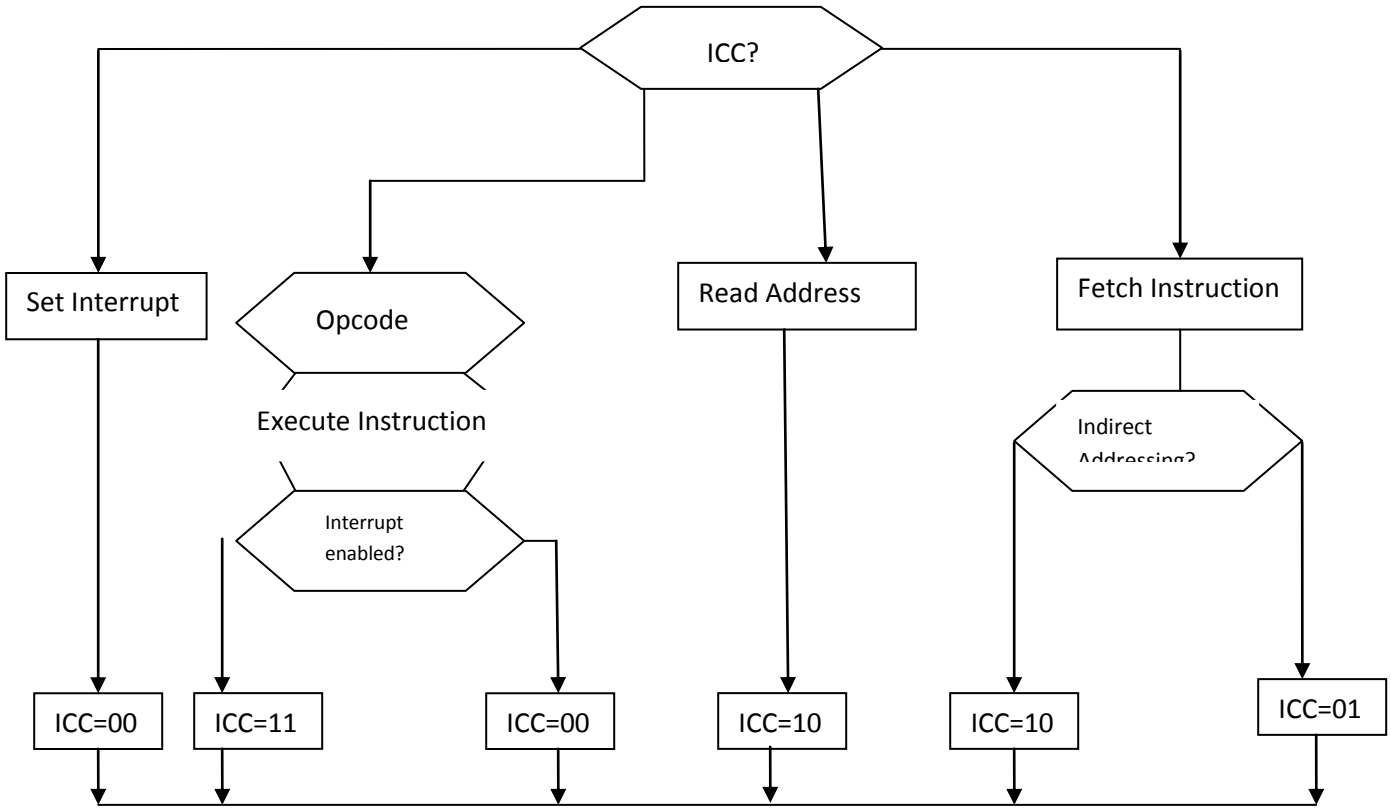


Fig: Flowchart for instruction Cycle

Control of processor

Functional Requirements

Functional requirements are the functions that the control unit must perform. These requirements help in design and implementation of control unit. The following steps leads to characterization of the Control Unit:

- Define the basic elements of CPU.
- Describe micro-operations that the CPU performs.
- Determine the functions that the Control Unit must perform to cause the micro-operations to be performed.

— **Basic functional elements:**

- ALU- Functional essence of computer.
- Registers- Store data internal to processor.
- Internal data paths- For data transfer between registers and register and ALU.
- External data paths- Links registers to memory and I/O modules through system bus.
- Control Unit- Causes operations to be happen within processor.

— **Micro-operations:**all the micro operations fall into the following categories-

- Transfer data from one register to another.

- Transfer data from a register to I/O.
- Transfer data from I/O to a register.
- Perform an arithmetic or logic operations using registers for input and output.

--- Functions →two basic functions performed by control units are –

- *Sequencing* – causes CPU to step through series of micro-operations in proper sequence
- *Execution* – causes each micro-operation to be performed.

Control Signals- Control unit operates on control signals. They must have logics for sequencing and execution.

- the control signals input or output to/from CU are-
  - *Clock:* CU causes one micro-operation to be performed in one clock pulse called cycle time.
  - *Instruction Register:* give information of which micro-operations to perform in execute cycle.
  - *Flags:* used in execution of conditional branch instructions.
  - *Control signals from control bus:* CU receives control signals such as interrupt, acknowledge etc.
  - *Control signals within CPU:* cause data movement, enable ALU etc.
  - *Control signals to Control bus:* for memory, I/O device etc.

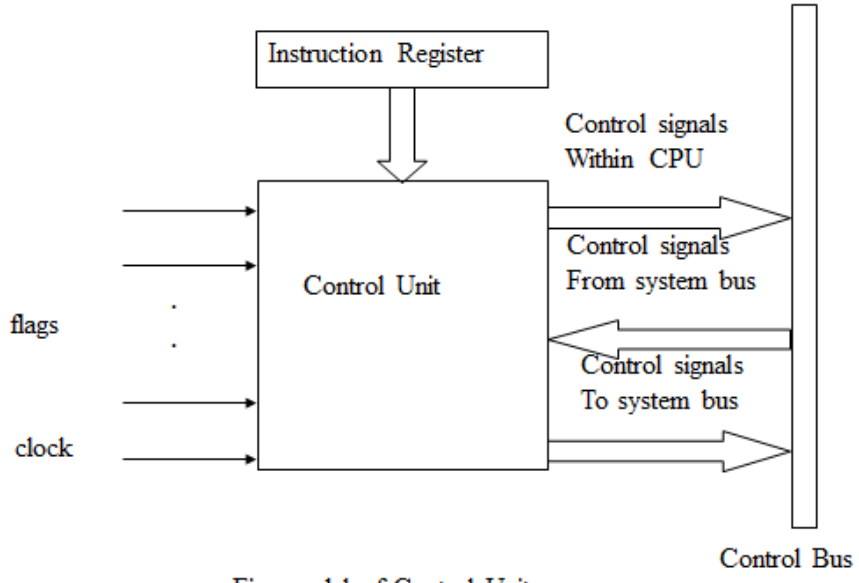


Fig. model of Control Unit

Control Signal example

Let us illustrate with examples

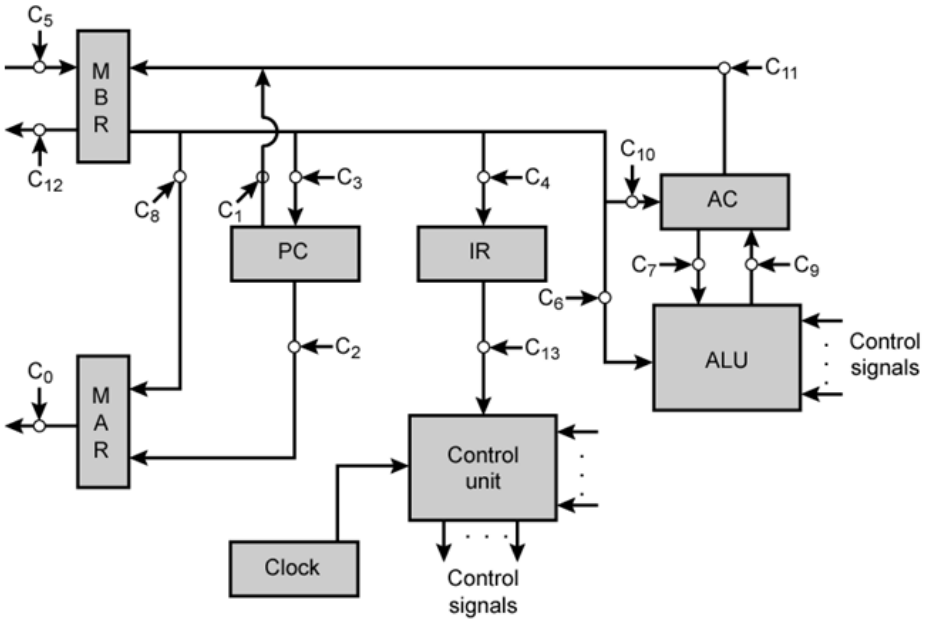


Fig: Data paths and Control Signals.

This is a simple processor with single accumulator. The data paths are indicated. The control signals from control unit are shown with termination of signals and labeled with C_i . Control unit receives signals/input from IR, Clock and Flags. With each clock cycle control unit reads all of its inputs and exits set of control signals. The destination of control signals are:

- Data paths: Control the internal flow of data .Eg in fetch cycle contents of MBR are transferred to IR.
- ALU- It controls the operation of ALU by a set of control signals. These signals activate various logic devices and gates with in ALU.
- System bus- The control unit sends control signals to the control lines of system bus. Eg memory READ.

Micro-operation and control signals

Micro-operations	Timing	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

Hardwired Implementation

Control unit can be implemented in two ways:

1. Hardwired Implementation
2. Micro-programmed Implementation

In hardwired implementation control unit is combination of circuit. Its input signal is transferred into set of output signals, which are control signals.

Control Unit Inputs:

The inputs of control signals are IR, clock, flags and control bus signals have some meaning. IR and clock are not directly useful for control unit. Control unit makes use of opcode and perform different action for different instruction. There is a unique logic input for each opcode. This function can be performed by a decoder which takes an encoded input and produce a single output. A decoder has n binary inputs and 2^n binary outputs. Each of 2^n different input patterns will activate a single unique output.eg.

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig; A decoder with four inputs and sixteen outputs

The clock portion of control unit issues a repeatative sequence of pulses. This is useful for measuring the duration of micro-operations. The control unit exits different control signals at different time units within a single instruction cycle. A counter is needed as ns input for different control signals which can be initialize after the end of an instruction cycle.

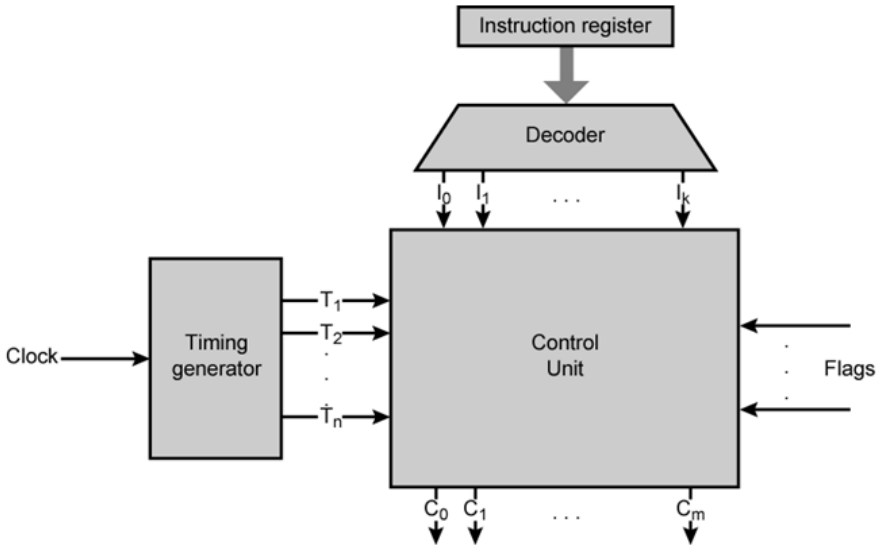


Fig:Control unit with Decoded input

Control unit logic

Let us discuss internal logic of control unit that produces output control signals as a function of input signals. A Boolean expression is derived for each inputs.

Let us consider a single control signal C5 from micro operations of instruction cycle. C5causes data to be read from memory into MBR. Let us consider two new control signals. Pand Q with following interpretation

P.Q= 00 Fetch Cycle

P.Q= 01 Indirect Cycle

P.Q= 10 Execute Cycle

P.Q= 11 Interrupt Cycle

Then for C5

C5=P'.Q'. t2+P'.Q. t2

This signal is asserted during second time unit of fetch and indirect cycle. C5 is also needed in execute cycle. For e.g. let us assume there is only two commands that read from memory ADD, ISZ then now C5 becomes

C5=P'.Q'. t2+P'.Q. t2 +P.Q'.(ADD+ISZ) .t2

This process is repeated for every control signal generated by processor. This Boolean equations defines the logic of control unit. These equations need a no. of combinational circuits. Thus microprogramming approach is used.

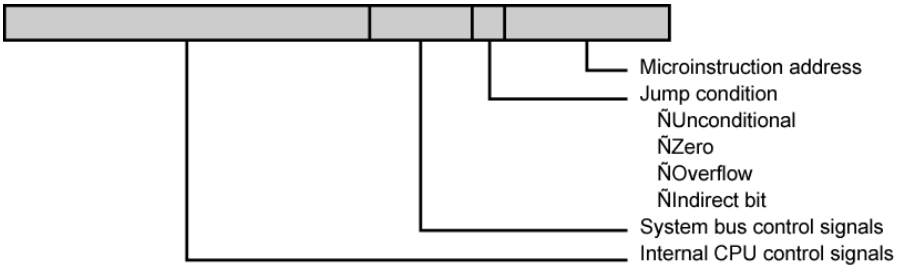
Micro programmed Control unit

Format of micro instruction

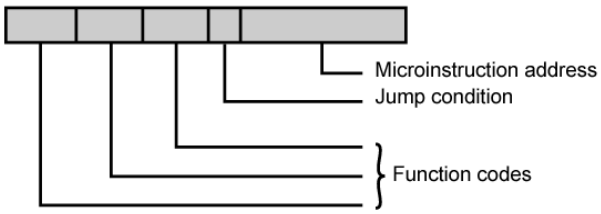
In micro-programmed control unit, in addition to use of control signals, each micro-operation is described in symbolic notation. This notation looks like programming language called **micro-programming.**Each lines of micro-programming describes the set of micro-operations occurring

at a time called **micro-instruction**. Sequence of micro-instructions is known as **micro-program** or **firmware**.The memory where micro-instructions are stored, is called **micro-code memory** or **control memory**. For any micro operations, control lines from control unit are either on or off. This can be represented by binary digit for each control line. Thus each micro operation is represented by patterns of 1s and 0s in control word.

All the control words are sequence of micro operations and we bring it together. Since micro operations are not fixed (sometimes indirect and sometimes direct) we put the control words in memory with unique address. Now an address field is added in control word for next control word to be executed for certain true condition. The true condition is also specified by few bits. This is known as horizontal micro-instruction.



(a) Horizontal microinstruction



(b) Vertical microinstruction

The format of microinstruction or control word is:

- One bit for each internal processor control line and one bit for each system bus control line.
- A conditional field that indicates condition for branching.
- A field with address of micro-instruction to be executed when branch is taken.

Such microinstruction is interpreted as:

1. To execute micro- instruction, turn on all the control lines indicated by 1, off all the control lines indicated by 0. The resulting signals will cause micro-operations to be performed.
2. If condition indicated by condition bit is false the next micro instruction is executed that is in sequence.
3. If condition bit is true, the next microinstruction is executed indicated in address field.

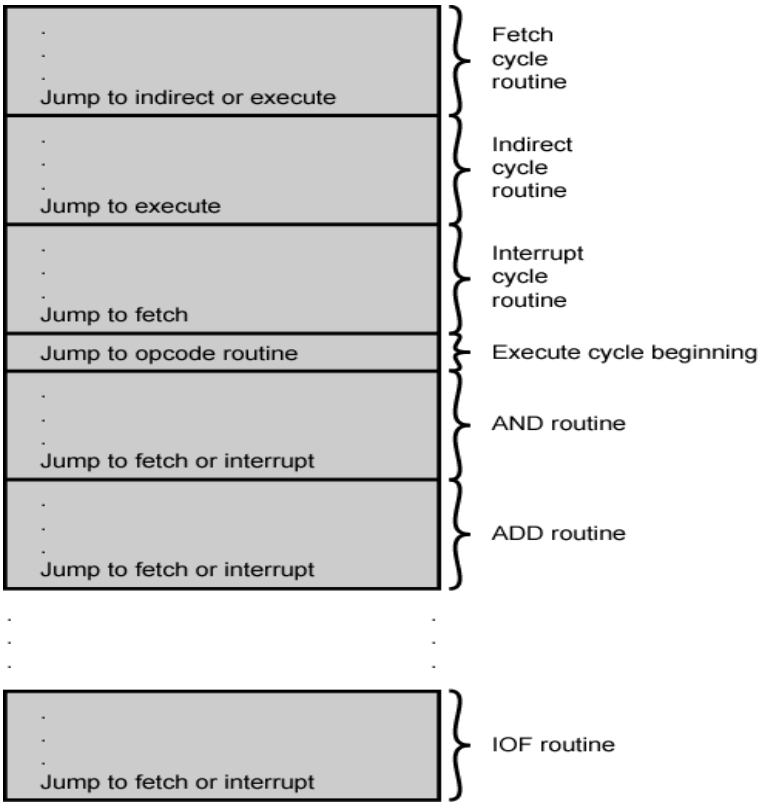


Fig: Organization of Control Memory

Micro programmed Control unit

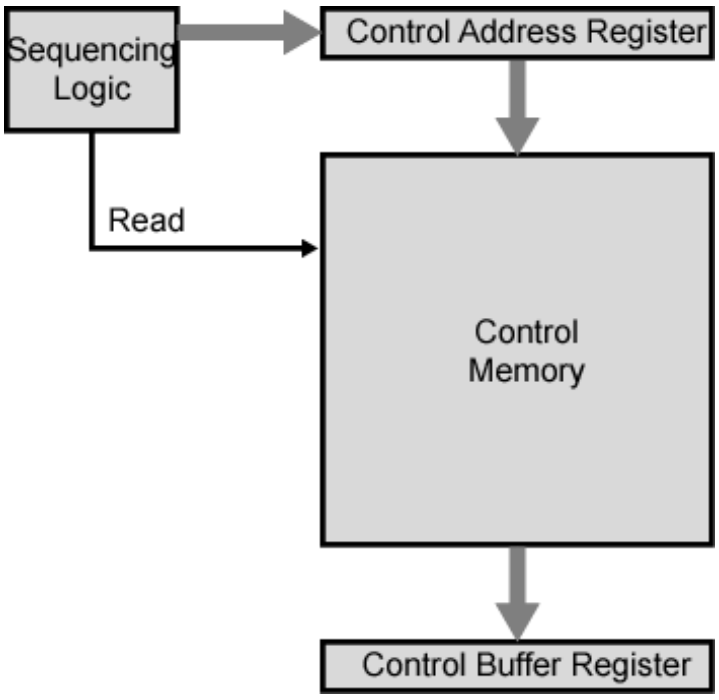


Fig: Control Unit Micro- Architecture

Here the set of micro-instruction is stored in control memory. The control address register contains the address of next instruction to be read. When a micro- instruction is read from the memory, it is transferred to control buffer register. The sequencing unit loads the control address register and issues a read command.

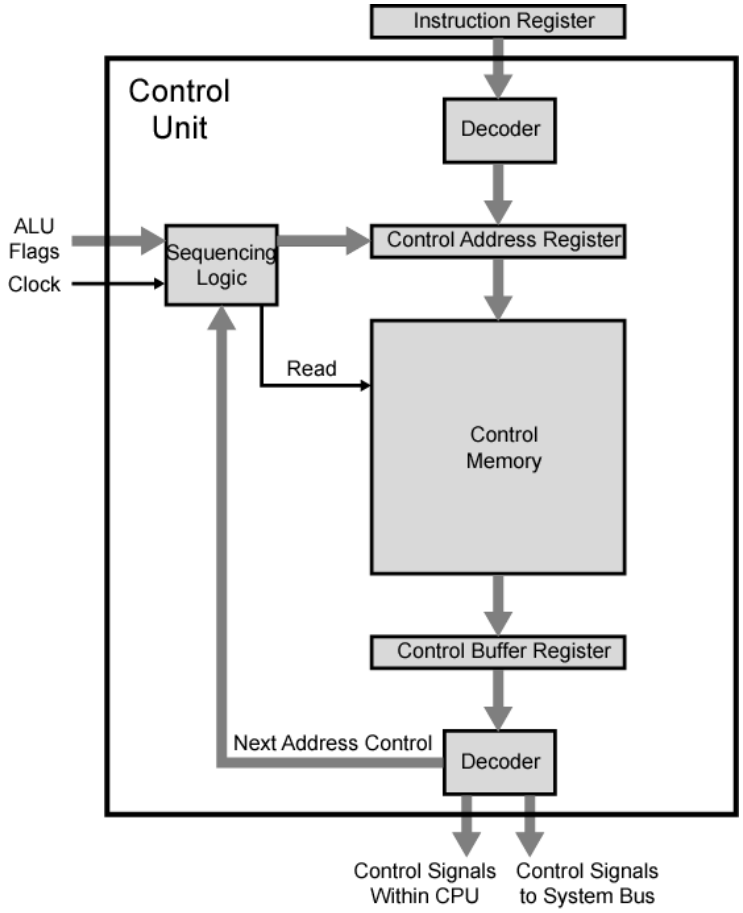


Fig: Functioning of Micro programmed Control Unit

The control unit functions as follows in a single clock pulse –

- To execute an instruction, the sequencing logic unit issues a READ command to the control memory.
- The word whose address is specified in the control address register is read into the control buffer register.
- The content of the control buffer register generates control signals and next address -information for the sequencing logic unit.
- 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

One of three decisions is made depending on the value of flags and CBR.

- Get the next instruction.
- Jump to a new routine based on jump microinstruction

- Jump to a machine instruction routine.

Micro – instruction Sequencing:
Micro programmed control unit performs two basic tasks:

1. Micro instruction Sequencing- Get the next instruction from control memory.
2. Micro instruction Execution- Generate control signals needed to execute the microinstruction.

Design consideration for sequencing

Two concerns are involved in design of micro-instruction sequencing technique.

- Size of micro-instruction.
- Time of address generation.

Address of next micro-instruction to be executed is in one of the following categories:

- Determined by IR (mapping function).
- Next sequential address (Next Address= Current Address +1).
- Branch.

The first category occurs only once per instruction cycle. Next sequential address is common in most designs. Branches both conditional and unconditional are necessary part of microprogram.

Sequencing Techniques

Current micro-instruction determines condition flag and content of IR, control memory address is generated for next micro-instruction. Based on the format of address information in micro-instruction, the technique of micro-instruction sequencing are of different types. These formats are –

- Two address fields.
- Single address field.
- Variable format.

Two address field

This is the simplest approach. There is two address field in each microinstruction. A multiplexer is provided that serves as a destination for address field and instruction register. Based on address selection input multiplexer transmits either opcode or one of two addresses to the control address register (CAR). CAR is decoded to produce next microinstruction address. Address selection signals are provided by branch logic module whose inputs are from flags and control part of control buffer register.

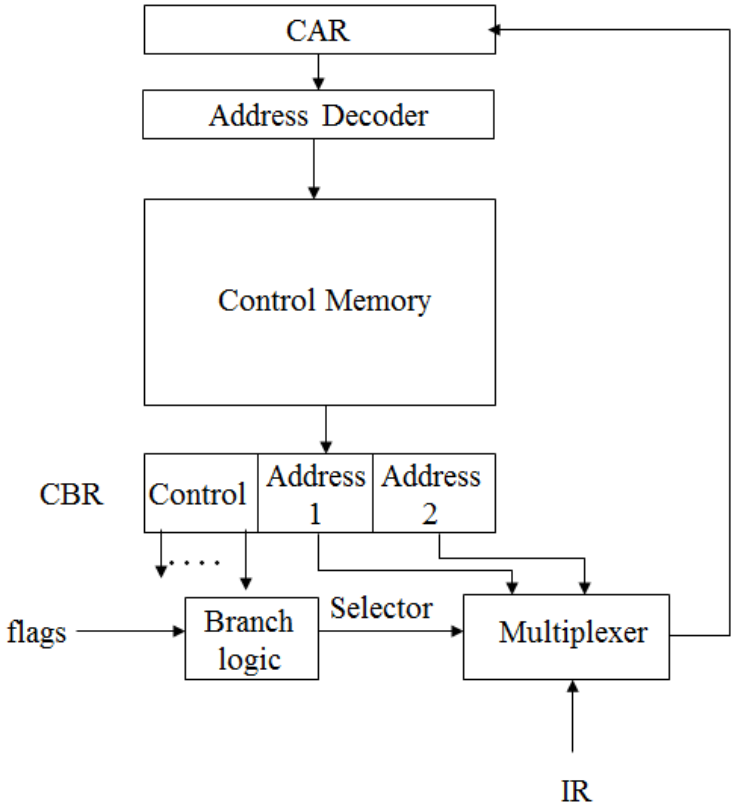


Fig: Two Address Field

Single address field

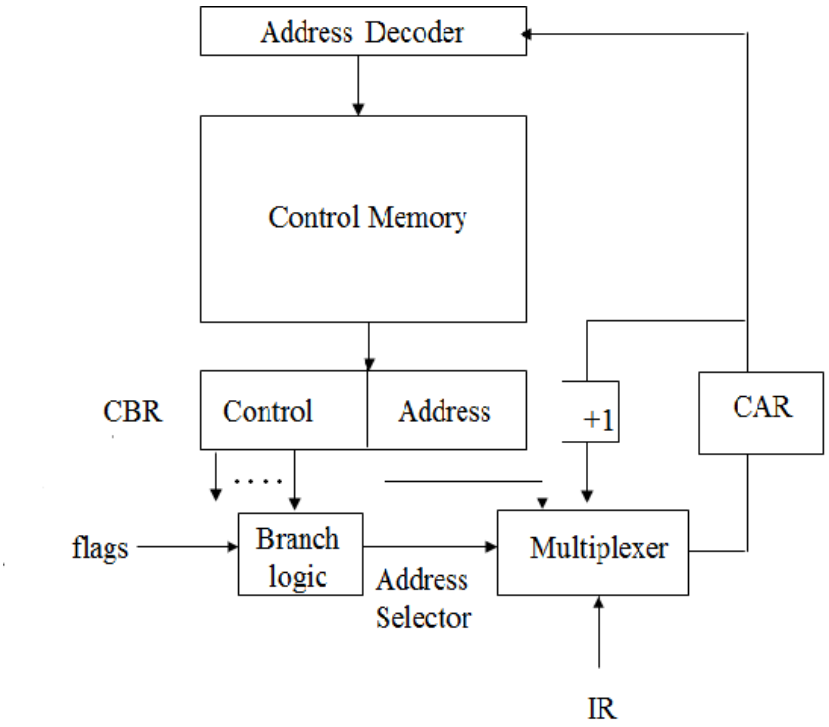


Fig: Single Address Field

Variable format

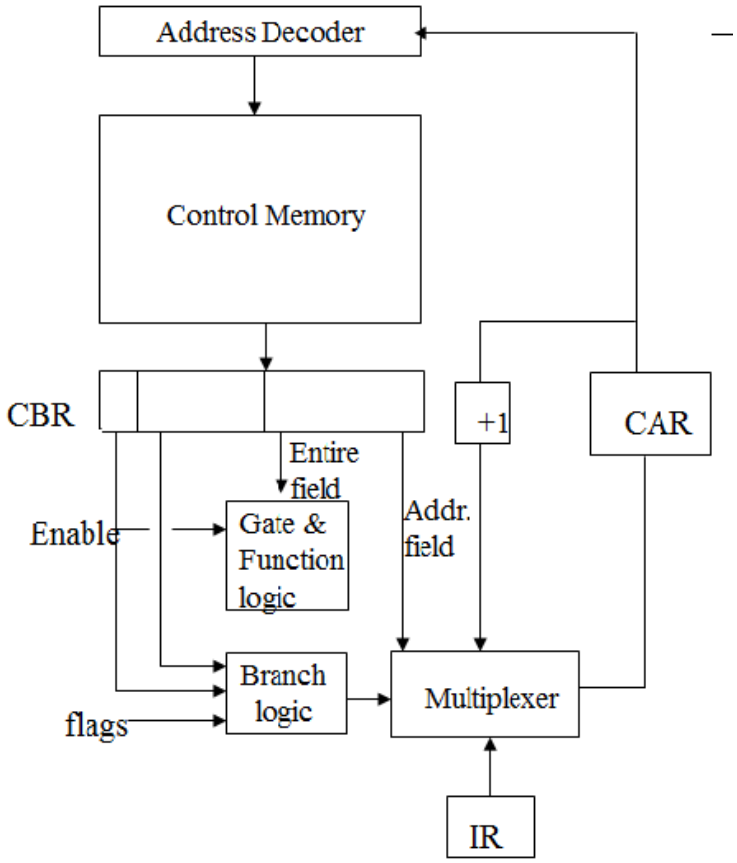


Fig: Variable Format

Another approach is to provide two different micro instruction format. Here one bit decides which format is being used. In one format remaining bits are used to activate control signals. In other format, some bits drive the branch logic and remaining bits provides the address. In first format next address is either next sequential address or an address by IR. In second format either a conditional or unconditional branch is specified. I this format one entire cycle is consumed with each branch instruction. In first format address generation is part of same cycle.

Address Generation

Address generation techniques are either explicit or implicit. In explicit addresses are available in micro instruction. In implicit additional logic is required to generate address.

Explicit techniques are dealt above. With two address field, two addresses are available with each microinstruction. With single and variable format branch instruction can be implemented. Conditional branch depends upon-ALU flags, part of opcode or address, parts of selected register like sign bit, status bit in control unit.

Implicit techniques are:

- Mapping
- Residual Control

Mapping is required with all designs. The opcode portion of a machine instruction must be mapped into microinstruction address. This occurs once per instruction cycle.

Another technique is combining or adding two portion of an address to form an complete address.e.g. IBM 3033.Here CAR is 13 bit long. The higher 8 bits normally do not change and ae copied directly form 8 bit field of an microinstruction and the remaining 5 bits are set to specify the specific addess of next microinstruction to be fetched.

00 07	08	09	10	11	12
BA(8) BB(4)	BC(4)	BD(4)	BE(4)	BF(7)	

Each of these bits is determined by 4 bit field except one with 7 bits.

The final technique is residual control. This involves the use of microinstruction address that has been previously saved in temporary storage within control unit. LSI-11 is its example.LSI-11 is a microcomputer version of PDP-11. It makes use of 22-bit microinstruction and a control memory of 2k 22 bit words. Next instruction is generated by one of these five ways:

1. Next Sequential Address-In absence of other instruction CAR is incremented by 1.
2. Opcode Mapping- At the beginning of each instruction cycle, the next microinstruction is determined by opcode.
3. Subroutine facility- A one level subroutine facility is provided. One bit in every micro instruction is dedicated to this task. When the bit is set an 11 bit return register is loaded with the updated contents of CAR. A subsequent microinstruction that specifies a return will cause the CAR to be loaded from return register.

Micro instruction Execution

Microinstruction execution is to generate control signals. Some of these signals are internal and remaining to external control bus or interface.

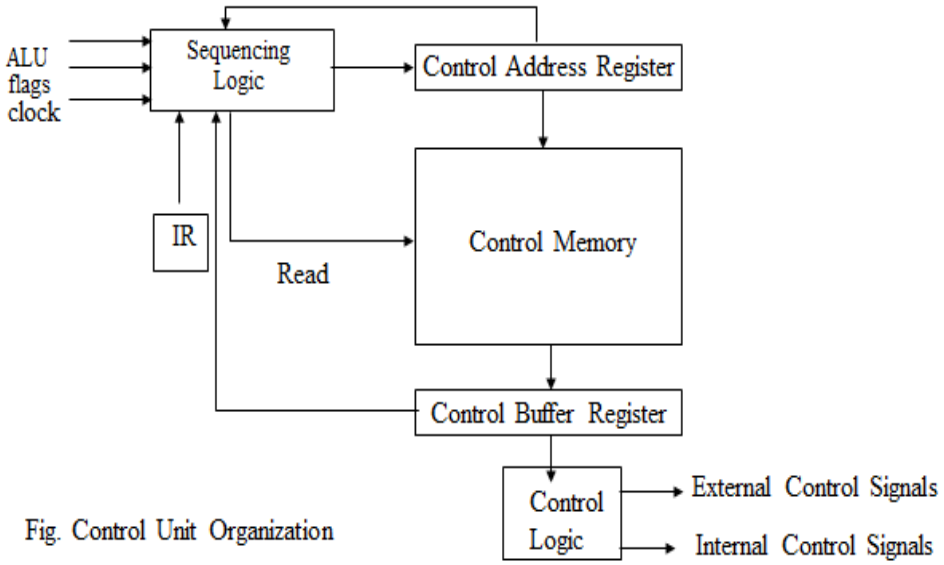


Fig. Control Unit Organization

The major module in this diagram is sequencing logic and control logic. Sequencing logic generates the address of next microinstruction taking inputs the IR, ALU flags, clock, CAR (for incrementing) and CBR.

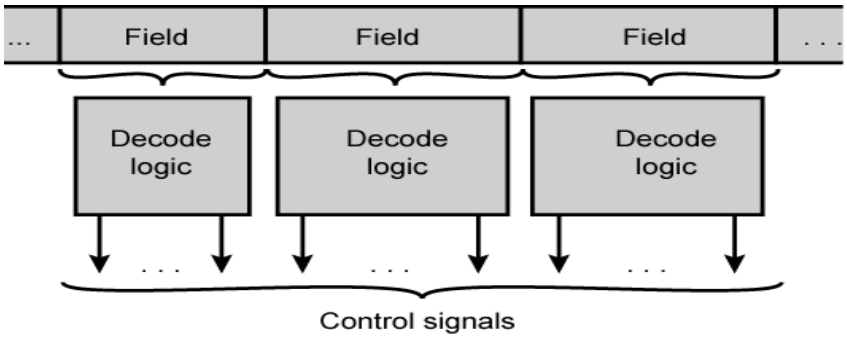
The control logic generates control signals. The format and content of microinstruction determines the logic of control logic.Microinstructions can be classified in the following ways –

— Vertical/Horizontal- It is related to width of microinstruction. The vertical microinstruction have length in the range of 16 to 40 bits and horizontal in the range of 40 to 100 bits.

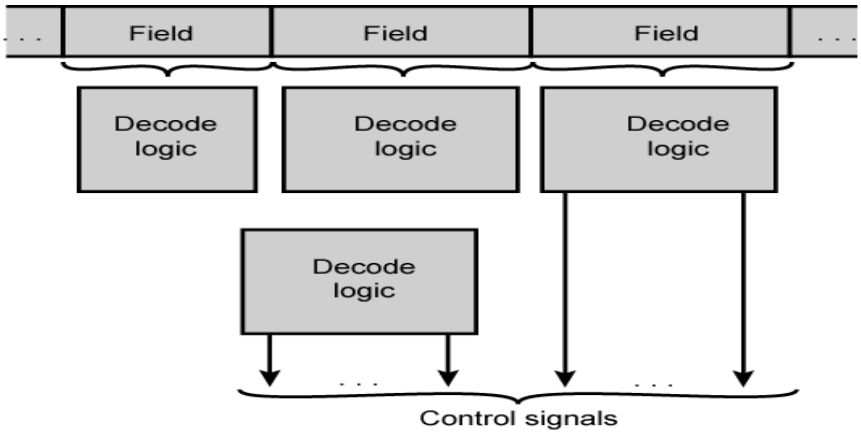
— Packed/unpacked- The degree of packing relates to the degree of identification between a given control task and specific microinstruction bits. As the bits become more packed, it contains more information.

— Hard/soft microprogramming- It suggests degree of closeness to the underlying control signals and hardware layout. Hard micro programs are fixed and committed to read-only memory. Soft micro programs are changeable and are used for user microprogramming.

— Direct/indirect encoding– For encoding microinstruction is organized as a set of fields. Each field contains a code which upon decoding activates one or more control signals.



(a) Direct encoding



(b) Indirect encoding

When the microinstruction is executed, every field is decoded and generates control signals. Thus, with N fields n simultaneous actions are specified. Each the action results in the activation of one or more control signals.

Now considering individual field. A field consists of L bits can contain one of 2^L codes, each of which can be encoded a different control signal pattern. Only one code can appear in a field at a time thus the codes are mutually exclusive and the action they cause are also mutually exclusive.

The design of encoded microinstruction format can be stated as:

- Organize the field into independent fields.
- Define each field such that the alternative actions that can be specified by the field are mutually exclusive.

Two approaches can be taken to organize the encoded microinstruction into fields: function and resource. The functional encoding method identifies the function within the machine and decides the field by function type. For e.g.data transferring to Accumulator can be specified in one field. 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)

In indirect encoding one field is used to determine the interpretation of another field. E.g. consider an ALU that perform 8 different arithmetic operations and 8 different shift operations. 1-bit field could be used to indicate shift or arithmetic and a 3-bit field to indicate operation. This technique implies two levels of decoding increasing the propagation delays.

Application of Microprogramming

- The first introduction of micro-programming was in 1971.
- Set of current applications for micro-programming includes:
  - Realization of computers
  - Emulation
  - Operating system support.
  - Realization of special purpose devices.
  - High-level programming support.
  - Micro diagnostics.
  - User tailoring

i) Realization of computers:
   - Micro-programming offers a systematic technique for control unit implementation.

ii) Emulation:
   - *Emulation* is use of a micro-program on one machine to execute programs originally written for another.
   - e.g. system/360 could run old binary programs for IBM 1401 and 7094 without modification.

iii) Operating system support
   - Micro-programs can be used to implement primitives that replace important portions of operating system software.

iv) Realization of special purpose device
   - It may be used as vehicle for implementing special purpose devices that may be incorporated into a host computer.
   - E.g. TV card.

v) High-level language support
   - Various functions and data types can be implemented directly in firmware which is easier to compile into machine language.

vi) micro-diagnosis:
   - Micro-diagnostics is a feature which is used in diagnosis of microelectronics system.
   - This feature may be used to support monitoring, detection, isolation and repair of system errors.
   - E.g. if multiplying unit is malfunctioning, micro-programmed multiplier can take over.

vii) User tailoring
   - A number of machines provide a *writable control store* , implemented n RAM, which allow the user to write micro-programs. This allows the user to use the machine to the desired application.