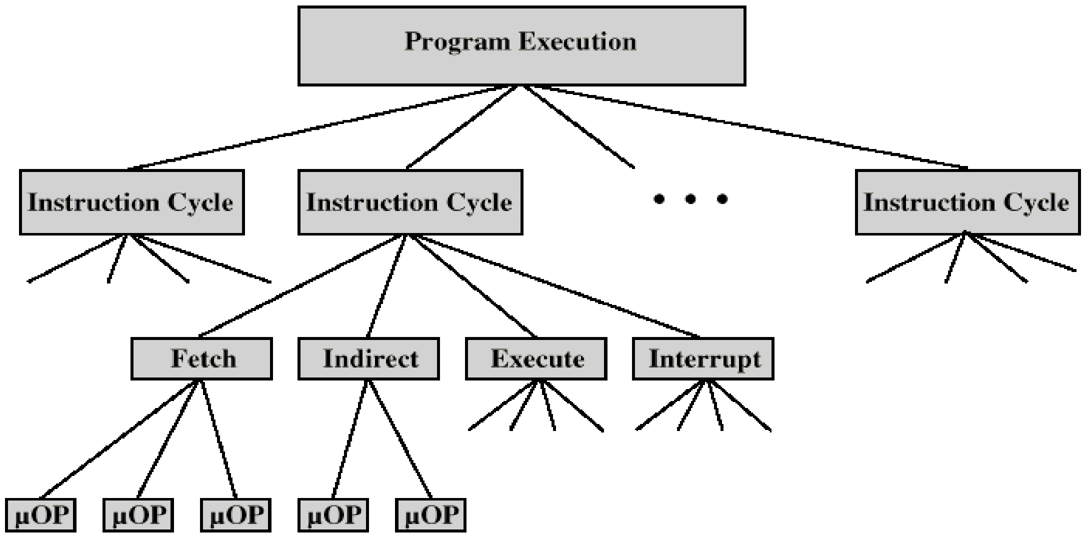
CA - MODULE –IV Control Unit Organization

1. MICRO-OPERATIONS

- The execution of an instruction involves the execution of a sequence of substeps, generally called cycles.
- For example, an execution may consist of fetch, indirect, execute, and interrupt cycles.
- Each cycle is in turn made up of a sequence of more fundamental operations, called micro-operations.
- A single micro-operation generally involves a transfer between registers, a transfer between a register and an external bus, or a simple ALU operation.
- Micro-operations are the functional, or atomic, operations of a processor



1.1. The Fetch Cycle

- Four registers are involved in fetch cycle.
    - o **Memory address register (MAR):** Is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation.
    - o **Memory buffer register (MBR):** Is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from memory.
    - o **Program counter (PC):** Holds the address of the next instruction to be fetched.
    - o **Instruction register (IR):** Holds the last instruction fetched.

- Fetch Sequence
    - o Address of next instruction is in PC
    - o Address (MAR) is placed on address bus
    - o Control unit issues READ command
    - o Result (data from memory) appears on data bus
    - o Data from data bus copied into MBR
    - o PC incremented by 1 (in parallel with data fetch from memory)
    - o Data (instruction) moved from MBR to IR
    - o MBR is now free for further data fetches

$$\begin{aligned}
t_1&: \mathrm{MAR} \leftarrow (\mathrm{PC})\\
t_2&: \mathrm{MBR} \leftarrow \mathrm{Memory}\\
&\ \mathrm{PC} \leftarrow (\mathrm{PC}) + I\\
t_3&: \mathrm{IR} \leftarrow (\mathrm{MBR})
\end{aligned}$$

1.2. The Indirect Cycle

- Once an instruction is fetched, the next step is to fetch source operands.
- If the instruction specifies an indirect address, then an indirect cycle must precede the execute cycle.

$$\begin{aligned}
t_1&: \mathrm{MAR} \leftarrow (\mathrm{IR(Address)})\\
t_2&: \mathrm{MBR} \leftarrow \mathrm{Memory}\\
t_3&: \mathrm{IR(Address)} \leftarrow (\mathrm{MBR(Address)})
\end{aligned}$$

- The address field of the instruction is transferred to the MAR.
- This is then used to fetch the address of the operand.
- Finally, the address field of the IR is updated from the MBR, so that it now contains a direct rather than an indirect address.

1.3. The Interrupt Cycle

- At the completion of the execute cycle, a test is made to determine whether any enabled interrupts have occurred.
- If so, the interrupt cycle occurs.

$$\begin{aligned}
t_1&: \mathrm{MBR} \leftarrow (\mathrm{PC})\\
t_2&: \mathrm{MAR} \leftarrow \mathrm{Save\_Address}\\
&\ \mathrm{PC} \leftarrow \mathrm{Routine\_Address}\\
t_3&: \mathrm{Memory} \leftarrow (\mathrm{MBR})
\end{aligned}$$

- In the first step, the contents of the PC are transferred to the MBR, so that they can be saved for return from the interrupt.
- Then the MAR is loaded with the address at which the contents of the PC are to be saved, and the PC is loaded with the address of the start of the interrupt-processing routine.
- The final step is to store the MBR, which contains the old value of the PC, into memory.
- The processor is now ready to begin the next instruction cycle.

1.4. The Execute Cycle

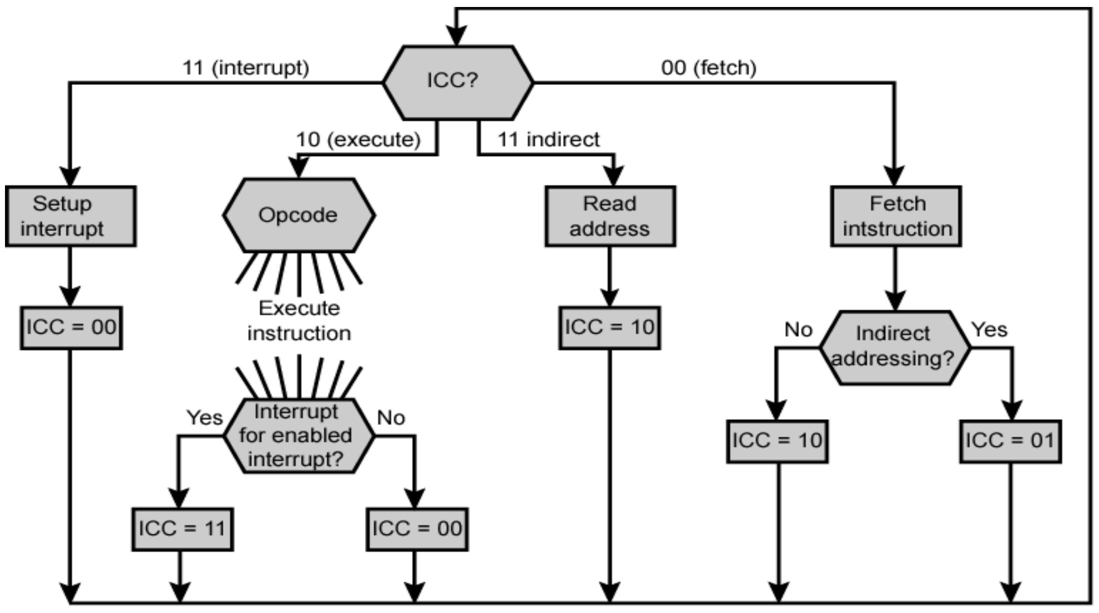
- Different for each instruction
  e.g. ADD R1,X - add the contents of location X to Register 1 , result in R1

```
t1:     MAR <- (IRaddress)
t2:     MBR <- (memory)
t3:     R1 <- R1 + (MBR)
```

1.5. The Instruction Cycle

- The instruction cycle can be decomposed into a sequence of elementary micro-operations.
- There is one sequence each for the fetch, indirect, and interrupt cycles, and, for the execute cycle, there is one sequence of micro-operations for each opcode.

- We assume a new 2-bit register called the *instruction cycle code* (ICC).
- The ICC designates the state of the processor in terms of which portion of the cycle it is in:
  - 00: Fetch
  - 01: Indirect
  - 10: Execute
  - 11: Interrupt
- At the end of each of the four cycles, the ICC is set appropriately.
- The indirect cycle is always followed by the execute cycle.
- The interrupt cycle is always followed by the fetch cycle.
- For both the fetch and execute cycles, the next cycle depends on the state of the system.



2. CONTROL OF THE PROCESSOR

2.1. Functional Requirements

- The *functional requirements* for the control unit is those functions that the control unit must perform.
  - Define the basic elements of the processor.
  - Describe the micro-operations that the processor performs.
  - Determine the functions that the control unit must perform to cause the micro-operations to be performed.

2.1.1. Basic Elements of Processor

- The basic elements of processor are :
  - ALU
  - Registers
  - Internal data pahs
  - External data paths
  - Control Unit
- The ALU is the functional essence of the computer.
- Registers are used to store data internal to the processor. Some registers contain status information needed to manage instruction sequencing (e.g., a

program status word). Others contain data that go to or come from the ALU, memory, and I/O modules.

- Internal data paths are used to move data between registers and between register and ALU.
- External data paths link registers to memory and I/O modules, often by means of a system bus.
- The control unit causes operations to happen within the processor.

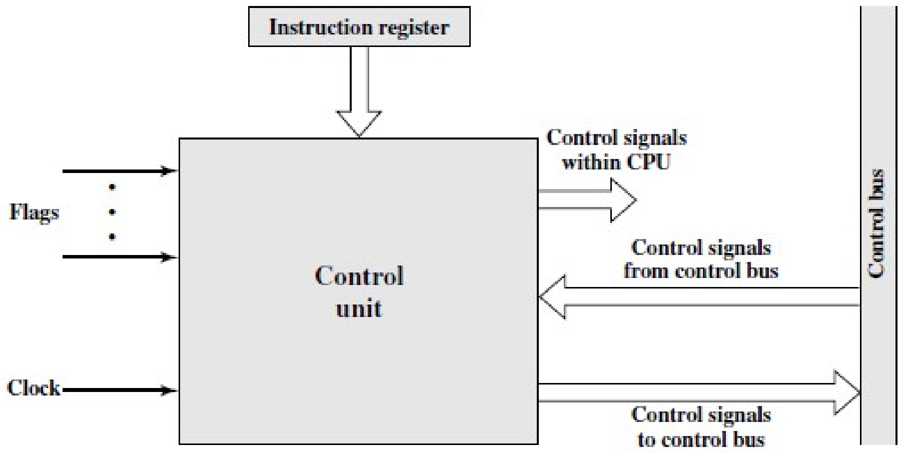
2.1.2. Types of Micro-operation

- Transfer data from one register to another.
- Transfer data from a register to an external interface (e.g., system bus).
- Transfer data from an external interface to a register.
- Perform an arithmetic or logic operation, using registers for input and output.

2.1.3. Functions of Control Unit

- **Sequencing:** The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.
- **Execution:** The control unit causes each micro-operation to be performed.
- This is done using Control Signals

2.2. Control Signals



Instruction register
Control signals within CPU
Flags
Control unit
Control signals from control bus
Control bus
Clock
Control signals to control bus

Figure 15.4 Block Diagram of the Control Unit

- For the control unit to perform its function, it must have inputs that allow it to determine the state of the system and outputs that allow it to control the behavior of the system

- The inputs are:
  - **Clock:** This is how the control unit "keeps time." The control unit causes one micro-operation to be performed for each clock pulse. This is also referred to as the processor cycle time.
  - **Instruction register:** The opcode and addressing mode of the current instruction are used to determine which micro-operations to perform during the execute cycle.
  - **Flags:** These are needed by the control unit to determine the status of the processor and the outcome of previous ALU operations.
  - **Control signals from control bus:** The control bus portion of the system bus provides signals to the control unit.
- The outputs are:

    - **Control signals within the processor:** These are two types: those that cause data to be moved from one register to another, and those that activate specific ALU functions.
    - **Control signals to control bus:** These are also of two types: control signals to memory, and control signals to the I/O modules.
- Three types of control signals are used:
    - those that activate an ALU function,
    - those that activate a data path,
    - those that are signals on the external system bus or other external interface

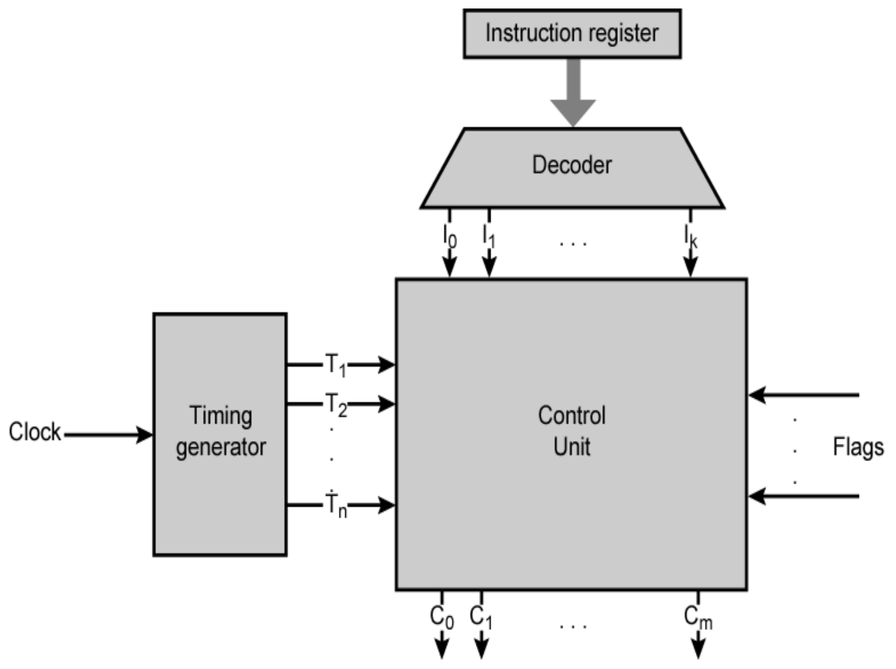
3. CONTROL UNIT

- Two types of control unit implementation
    - Hardwired implementation
    - Micro programmed implementation

3.1. Hardwired Implementation

- In a hardwired implementation, the control unit is essentially a state machine circuit.
- Its input logic signals are transformed into a set of output logic signals, which are the control signals.
- The key inputs are the instruction register, the clock, flags, and control bus signals.
- In the case of the flags and control bus signals, each individual bit typically has some meaning (e.g., overflow).
- Instruction register
    - Op-code causes different control signals for each different instruction
    - Unique logic for each op-code
    - Decoder takes encoded input and produces single output
    - n binary inputs and 2^n outputs

- The clock portion of the control unit issues a repetitive sequence of pulses. This is useful for measuring the duration of micro-operations. Essentially, the period of the clock pulses must be long enough to allow the propagation of signals along data paths and through processor circuitry.
- However, the control unit emits different control signals at different time units within a single instruction cycle.
- Thus, we would like a counter as input to the control unit, with a different control signal being used for and so forth.
- At the end of an instruction cycle, the control unit must feed back to the counter to reinitialize it at T1.



Problems With Hard Wired Designs

- Complex sequencing & micro-operation logic
- Difficult to design and test
- Inflexible design
- Difficult to add new instructions

4. MICROPROGRAMMED CONTROL UNIT

- 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 microprogram.
- A microprogram consists of a sequence of instructions in a microprogramming language.
- These are very simple instructions that specify micro-operations.

4.1. Microinstructions

- In addition to the use of control signals, each micro-operation is described in symbolic notation.
- This notation looks like a programming language, 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***.

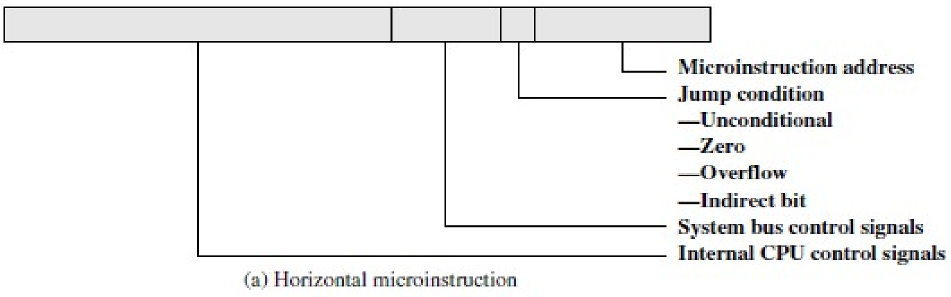
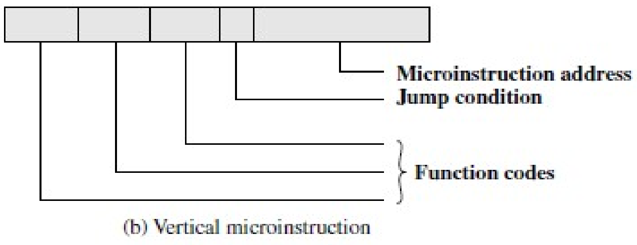
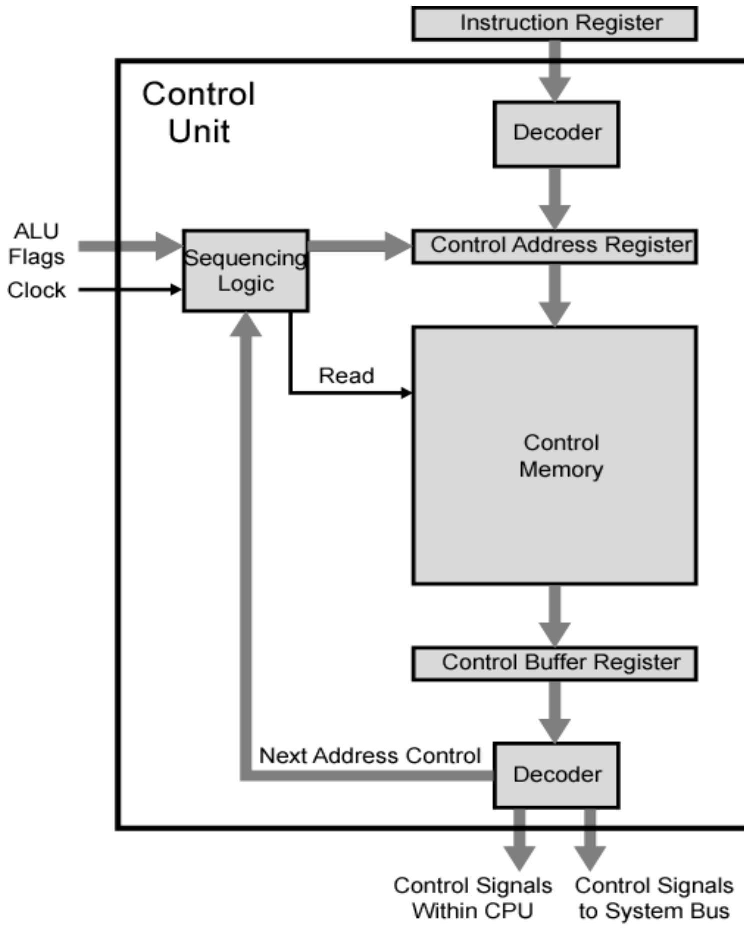

Microinstruction address
Jump condition
—Unconditional
—Zero
—Overflow
—Indirect bit
System bus control signals
Internal CPU control signals
(a) Horizontal microinstruction


Figure 16.1 Typical Microinstruction Formats

5. Functioning of Microprogrammed Control Unit

- The control memory contains a program that describes the behavior of the control unit.
- 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 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.
- The control unit functions as follows:
    - 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.

- 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.



Advantages and Disadvantages of Microprogramming

- Simplifies design of control unit
    - Cheaper
    - Less error-prone
- Slower

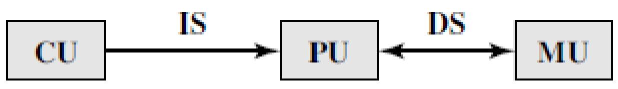
6. What is Parallel processing

- Parallel processing is a method of simultaneously breaking up and running program tasks on multiple microprocessors, thereby reducing processing time.
- Parallel processing may be accomplished via a computer with two or more processors or via a computer network.

7. Multiple processor organization

- A taxonomy first introduced by **Flynn** is still the most common way of categorizing systems with parallel processing capability.
- Flynn proposed the following categories of computer systems:
    - **Single instruction, single data (SISD) stream:**
    - **Single instruction, multiple data (SIMD) stream:**
    - **Multiple instruction, single data (MISD) stream:**
    - **Multiple instruction, multiple data (MIMD) stream:**

Single Instruction, Multiple Data Stream – SIMD



(a) SISD

- A single processor executes a single instruction stream to operate on data stored in a single memory.
- Eg : Uniprocessors.

Single instruction, multiple data (SIMD) stream

- A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis.
- Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors.
- Eg : Vector and array processors.

Multiple instruction, single data (MISD) stream

- A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence.
- This structure is not implemented.

Multiple instruction, multiple data (MIMD) stream

- A set of processors simultaneously execute different instruction sequences on different data sets.
- Eg : SMPs, clusters, and NUMA systems.