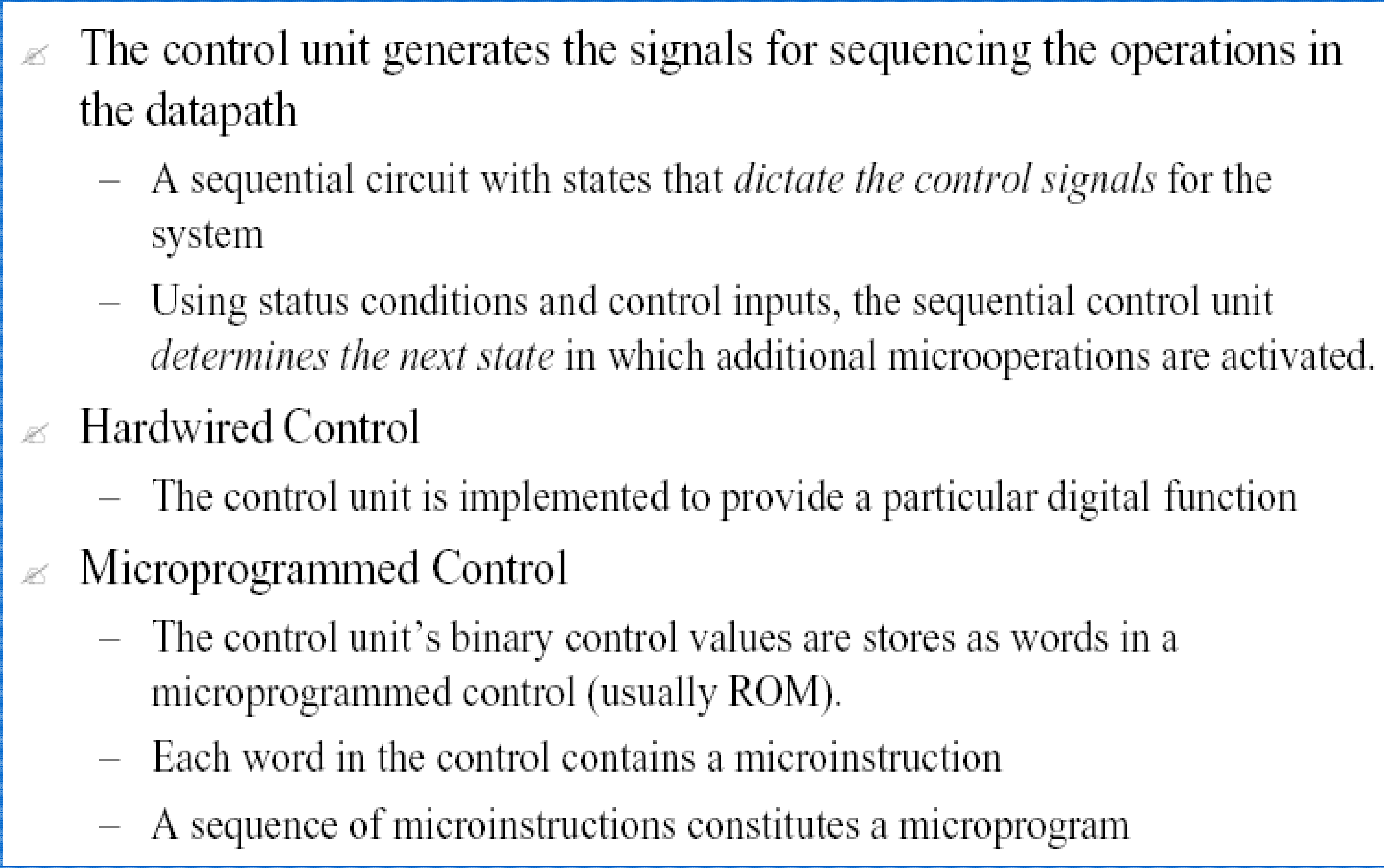
UNIT-III CONTROL UNIT DESIGN

- INTRODUCTION
- CONTROL TRANSFER
- FETCH CYCLE
- INSTRUCTION INTERPRETATION AND EXECUTION
- HARDWIRED CONTROL
- MICROPROGRAMMED CONTROL

INTRODUCTION

- Important component of CPU is the controller



- The control unit generates the signals for sequencing the operations in the datapath
  - A sequential circuit with states that *dictate the control signals* for the system
  - Using status conditions and control inputs, the sequential control unit *determines the next state* in which additional microoperations are activated.
- Hardwired Control
  - The control unit is implemented to provide a particular digital function
- Microprogrammed Control
  - The control unit's binary control values are stores as words in a microprogrammed control (usually ROM).
  - Each word in the control contains a microinstruction
  - A sequence of microinstructions constitutes a microprogram

TERMINOLOGY

Microprogram
- **Program stored in memory that generates all the control signals required to execute the instruction set correctly**
- **Consists of microinstructions**

Microinstruction
- **Contains a control word and a sequencing word**
  - **Control Word - All the control information required for one clock cycle**
  - **Sequencing Word - Information needed to decide the next microinstruction address**
- **Vocabulary to write a microprogram**

Control Memory(Control Storage: CS)
- **Storage in the microprogrammed control unit to store the microprogram**

Writeable Control Memory(Writeable Control Storage:WCS)
- **CS whose contents can be modified**
  - **-> Allows the microprogram can be changed**
  - **-> Instruction set can be changed or modified**

Dynamic Microprogramming
- **Computer system whose control unit is implemented with a microprogram in WCS**
- **Microprogram can be changed by a systems programmer or a user**

TERMINOLOGY

Sequencer (Micro program Sequencer)

A Micro program Control Unit that determines the Microinstruction Address to be executed in the next clock cycle

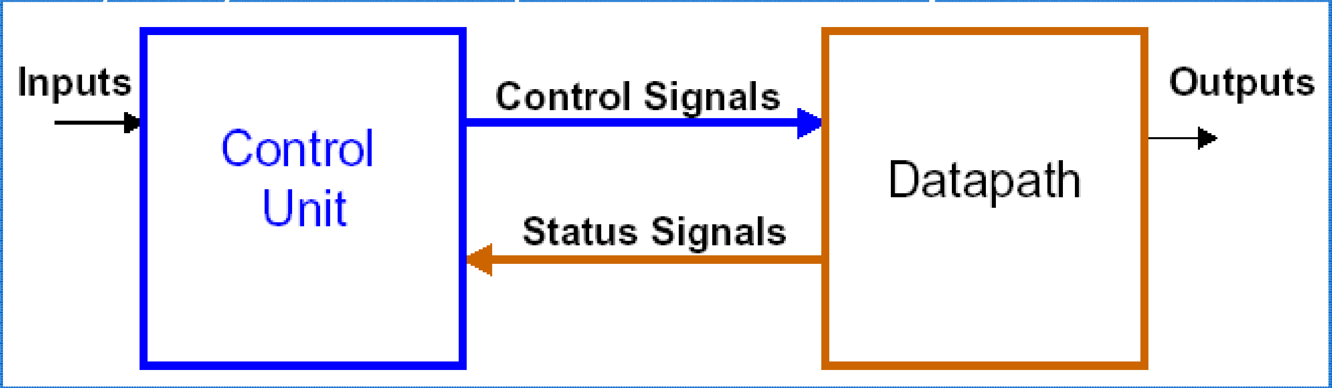
- In-line Sequencing
- Branch
- Conditional Branch
- Subroutine
- Loop
- Instruction OP-code mapping

Controller functions

1. Fetch and instruction sequencing (fetch cycle)-Generates control signal to fetch instruction from memory and the sequence of operations involved in processing an instruction
2. Instruction interpretation and execution (execution cycle)-Tasks involved are
   - » Interpreting the operand addressing mode implied in the operation code and fetching the operands
   - » Sequencing the successive micro operations on the data path to execute the operation code specified in the instruction
3. Interrupt processing (interrupt cycle)-Process unmasked interrupt. Tasks are
   - » Suspend execution of current program
   - » Save context
   - » Set PC to start address of interrupt handler routine
   - » Process interrupt
   - » Restore context and continue interrupted program

Interaction Between Data and Control Units

- Digital Systems can be partitioned in a **Datapath** and **Control Unit**



•Selects the microoperation
•Determines the sequence (based on status and input signals)
•Design: State diagram or ASM:
  •Microoperations
  •Sequence

•Registers
•MUXes, ALU, Shifters, Comb. Circuits and buses
•Implements microoperations (under control of the control unit)

Control Unit

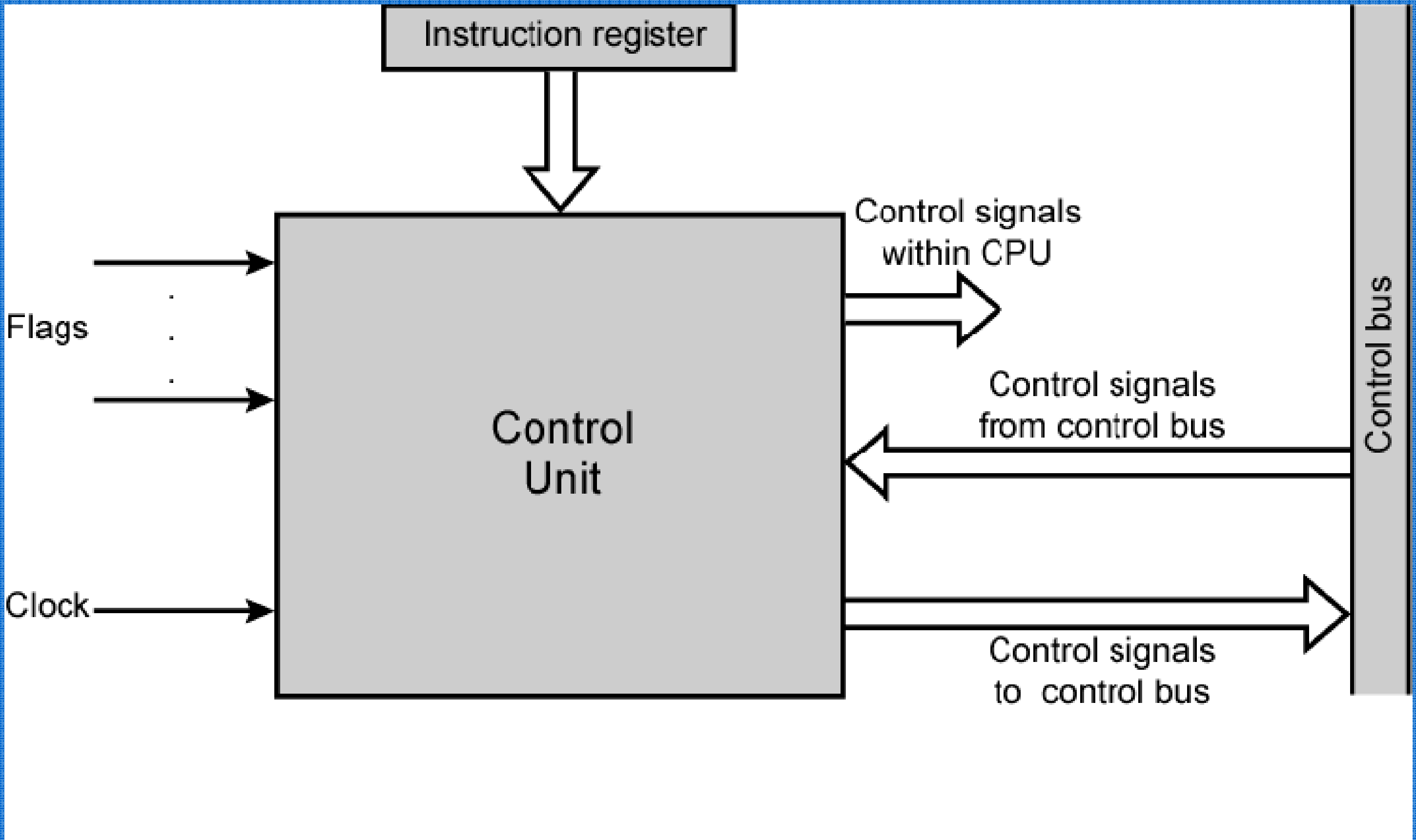

Instruction register
Control signals within CPU
Flags
Control Unit
Control signals from control bus
Control bus
Clock
Control signals to control bus

Figure 16.4 Block Diagram of the Control Unit

CONTROL TRANSFER

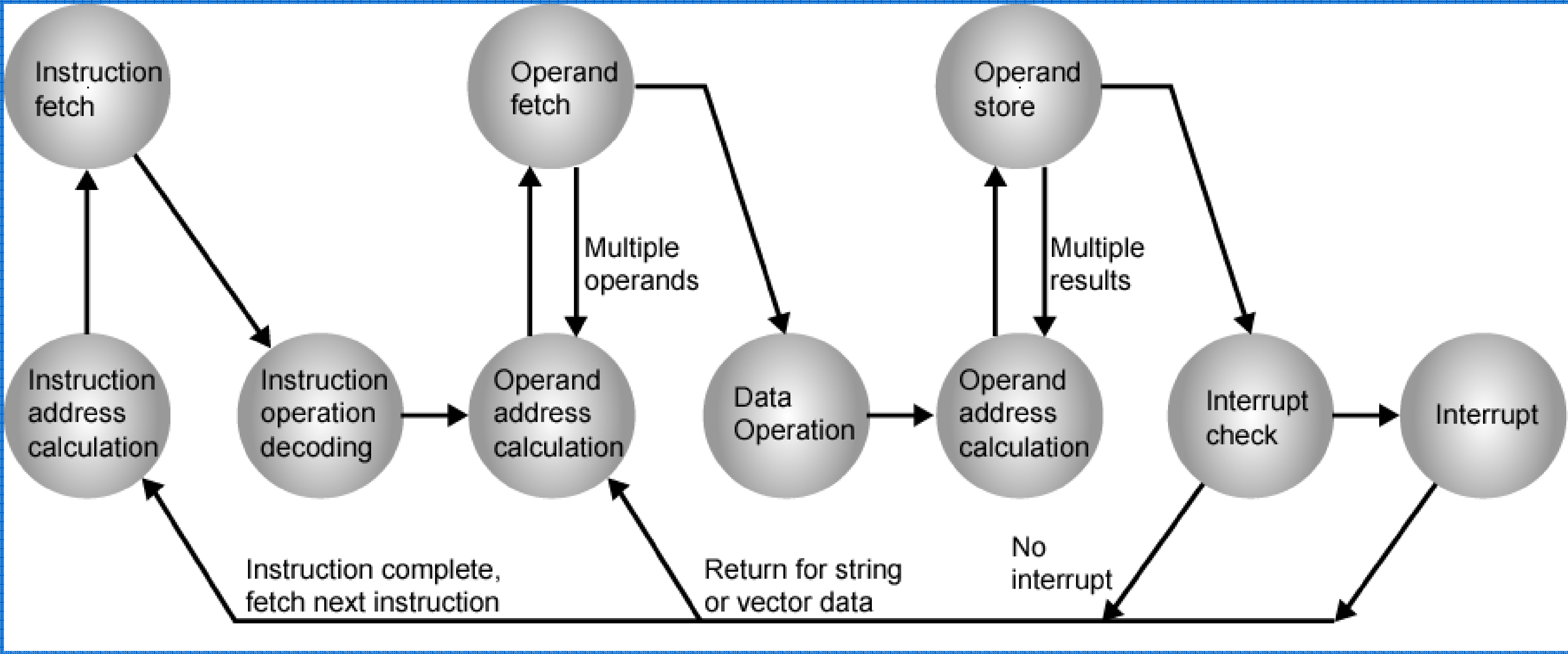
- CPU fetches and executes each instruction of the program
- Successively goes through fetch, execute and interrupt cycles
- Two flip-flops marked F and E identify each of these 3 cycles

- Two general operations
  - Instruction control transfer
  - Program control transfer

CPU Cycles

F	E	Comments
0	0	Not used
1	0	Fetch Cycle
0	1	Execute Cycle
1	1	Interrupt Cycle

Instruction Cycle (with Interrupts) - State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string or vector data
No interrupt

Instruction Control Transfer

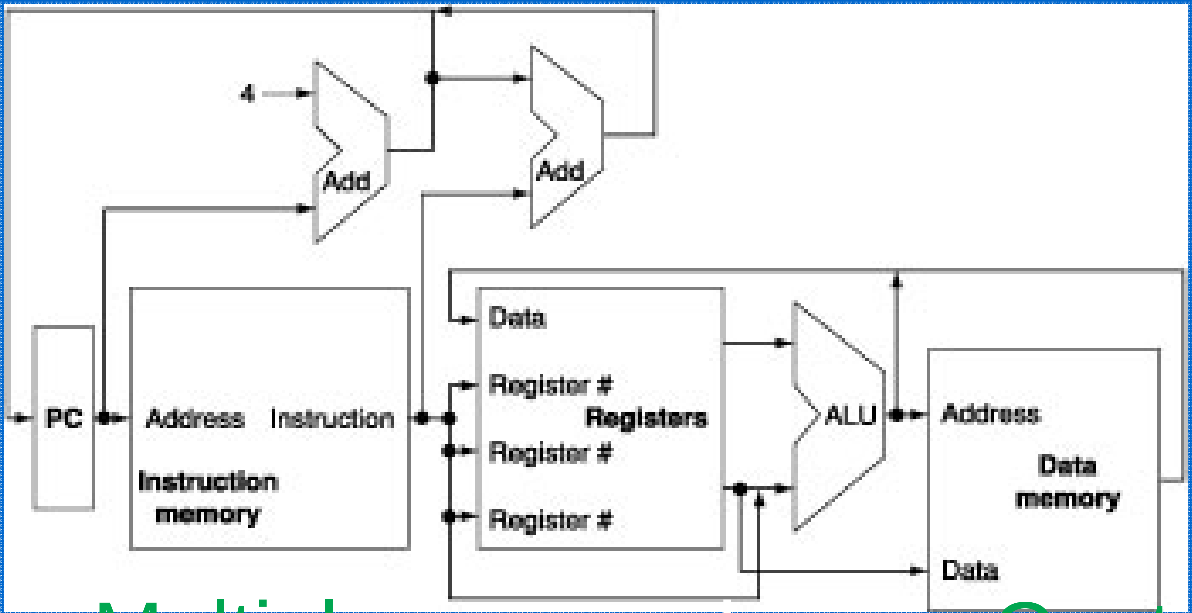
- PC holds the address of the instruction to be executed
- During fetch cycle PC is incremented to hold the address of the next instruction
- Assuming each instruction has a length of one word, then PC incremented by one, that is,PC `<= PC + 1;`
- Length of a variable length instruction needs to be specified in some bits of the operation code field of instruction
- Transfer of control to non sequential instruction occurs during execution cycle of conditions branch with specified condition satisfied or by an unconditional branch instruction

Example

- Jump X (Unconditional branch to X) or
- Jump ZX (Branch to X if the result of last arithmetic operation is zero)
- X stored in the operand field of instruction register

Instruction Control Transfer

- For most instructions: fetch instruction, fetch operands, execute, store.
- An abstract view of the implementation of the MIPS subset showing the major functional units and the major connections between them



- Missing Multiplexers, and some Control lines for read and write.

PROGRAM CONTROL TRANSFER

- While program P1 is running ,CPU is under control of P1
- Transfer of program control may occur in 2 situations

1. There is an instruction in P1 which calls a subroutine denoted as program P2

   JMPSUBX, where X denotes starting address of P2

2. While an instruction in P1 is getting fetched and executed ,an interrupt flag gets set, where X denotes starting address of ISR(P2).After execution of current instruction branch to location X takes place

- Prior to transferring control to P2 ,the CPU status of P1 is stored in stack (push and pop operation)

FETCH CYCLE

- The instruction whose address is determined by the PC is obtained from the memory
- Loaded into the IR.
- The PC is then incremented to point to the next instruction and switch over to execution cycle
- CPU enters fetch cycle if F=1 and E=0

C_{01}: MAR $\leftarrow$ PC

$C_{02}C_{03}$:MDR $\leftarrow$ M(MAR);PC $\leftarrow$ PC+1

C_{04}:IR $\leftarrow$ MDR

C_{05}:F $\leftarrow$ 0; E $\leftarrow$ 1

Sequence of micro operations of fetch cycle

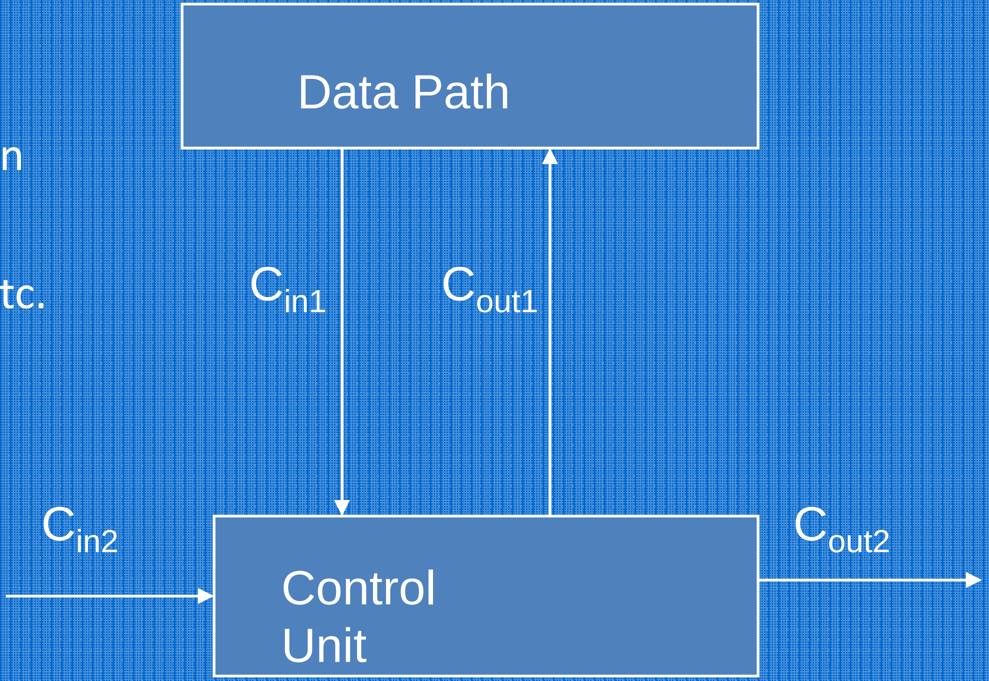
INSTRUCTION INTERPRETATION AND EXECUTION

- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions
  - Different for each instruction
- e.g. ADD X, R1 - add the contents of location X to Register 1 , result in R1
- C_{61}: MAR <- ($IR_{address}$)
- C_{62}: MDR <- (memory)
- C_{63}: R1 <- R1 + (MDR)
- Note no overlap of micro-operations

Input-Output behavior of a control unit

- Set of signals C_{in1} input to the controller from the datapath includes those from the operation code of IR, various flag registers of ALU,interrupt , etc. dictate the flow

  of control for the algorithm implemented on the datapath for an operation code
- Other input signals
  C_{in2} are those control
which control the operation
of the controller itself,
such as start, stop, clock, etc.
- C_{out1} fed at
appropriate
point on the datapath
to sequence and control
the micro operations
 associated with operation code

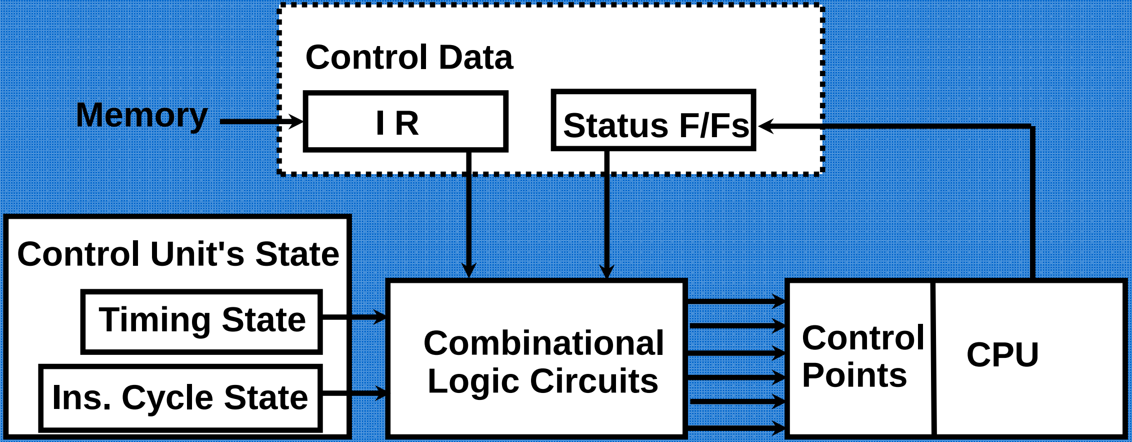


- Other control signal C_{out2} cover signals such as busy, operation complete, etc transmitted to other control units

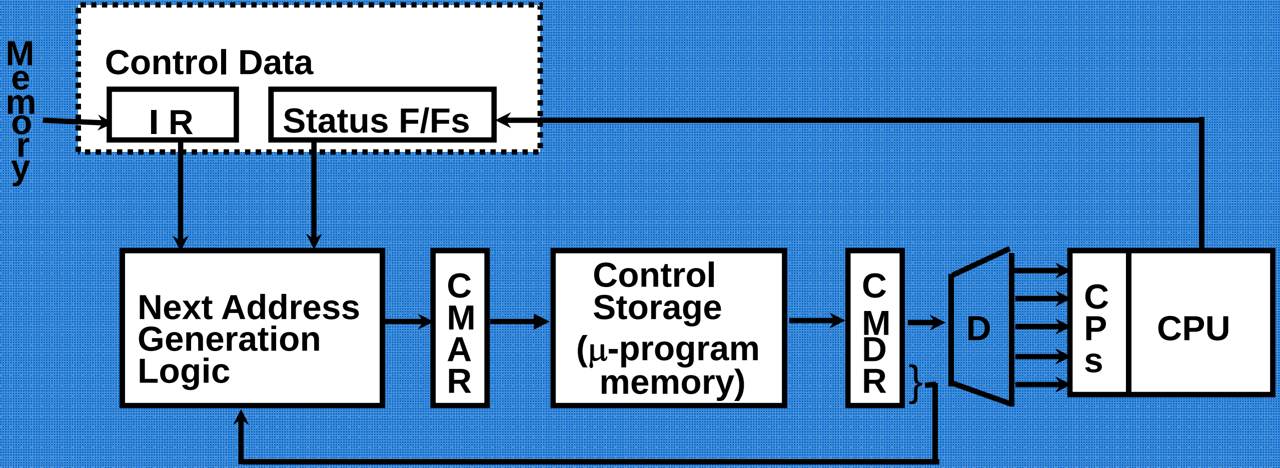
COMPARISON OF CONTROL UNIT IMPLEMENTATIONS

Control Unit Implementation

Combinational Logic Circuits (Hard-wired)



Microprogram


Memory
Control Data
I R
Status F/Fs
Next Address Generation Logic
CMAR
Control Storage (μ-program memory)
CMDR
D
CPs
CPU

HARDWIRED CONTROL

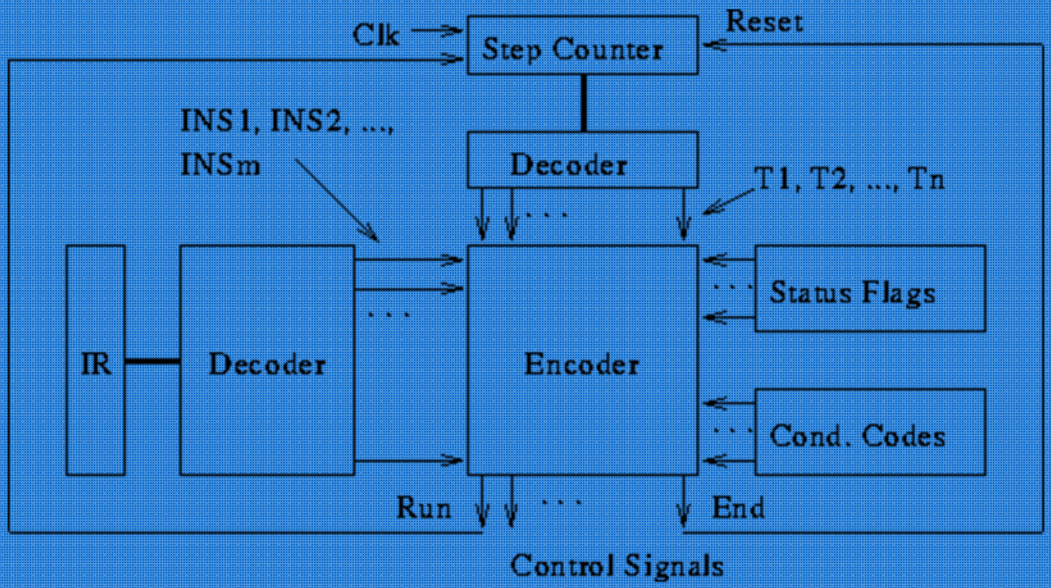
- The controller is designed as a sequential logic circuit which generates the specific sequence of control signals as its primary output

- The sequence of 4 control signals $C_{0,}$ $C_{1,}$ $C_{2,}$ C_3 can be developed by using a 2-bit sequence counter

  $C_0=C_{01}$: MAR $\leftarrow$ PC
  $C_1=C_{02}C_{03}$:MDR $\leftarrow$ M(MAR);PC $\leftarrow$ PC+1
  $C_2=C_{04}$:IR $\leftarrow$ MDR
  $C_3=C_{05}$:F $\leftarrow$ 0; E $\leftarrow$ 1
  Sequence of micro operations of fetch cycle

- Its output is decoded to derive the desired control signals in the given sequence.

Controller Block Diagram



Detailed control sequencing

- ADD (R3), R1 **High-level execution sequence:**
- Fetch instruction.
- Fetch memory operand.
- Perform addition.
- Store result in R1.

Detailed control sequencing:

1: PCout, MARin, Read, Clear Y, Set carry-in to ALU, Add, Zin

2: Zout, PCin, WMFC

3: MDRout, IRin

4: R3out, MARin, Read

5: R1out, Yin, WMFC

6: MDRout, Add, ZinZout, R1in, END

Methods for systematic design of hardwired control logic

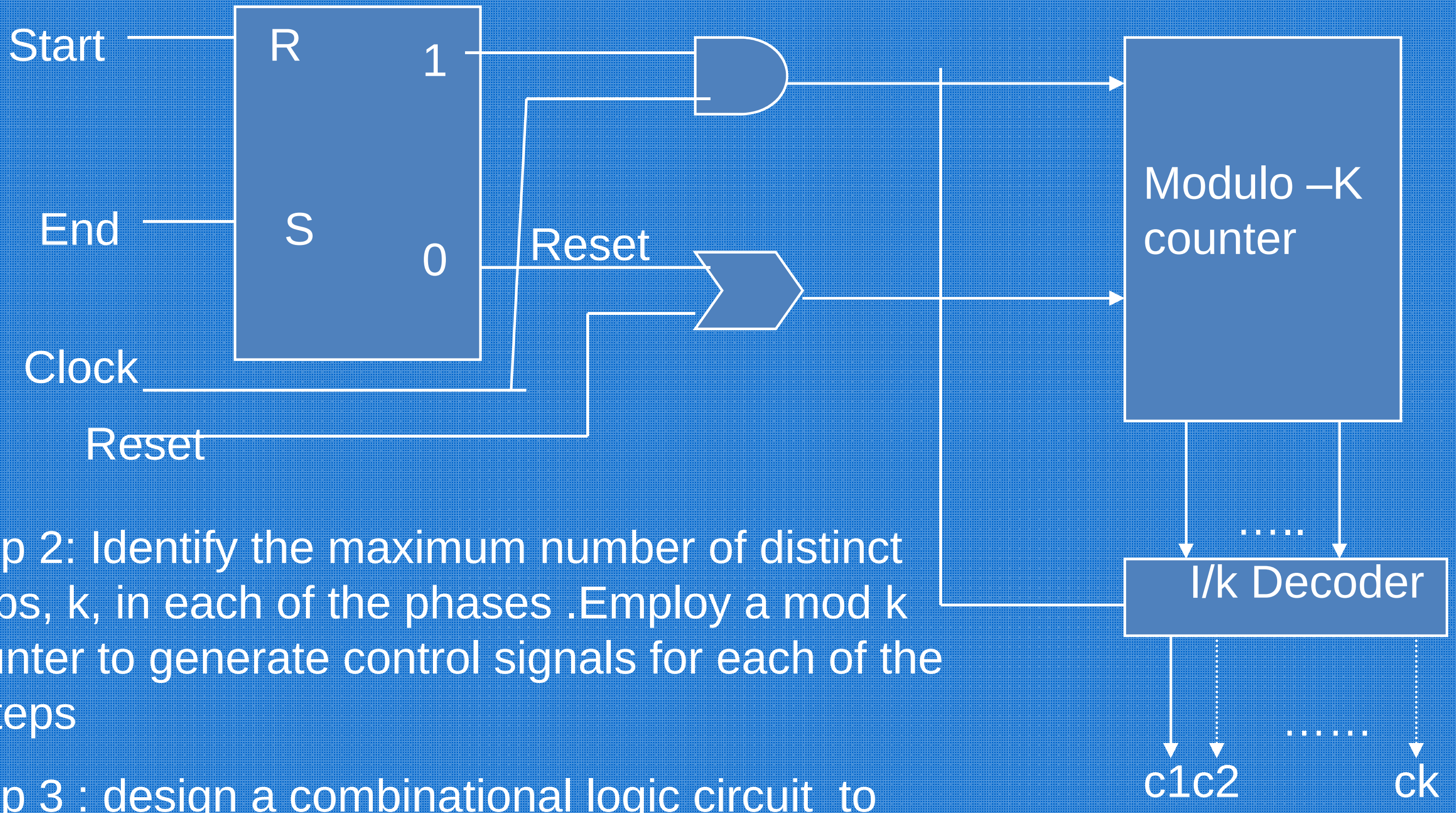
- Sequence counter method: To design controller of moderate complexity
- Delay element method: Depends on the use of clocked delay elements for generating the sequence of control signals
- State table method: Employs the algorithmic approach to sequential circuit design using classical state table method

Controller specification for the Fetch Sequence

- Address of next instruction is in PC (s01)
- Content of PC loaded into MAR (c01) ,s02
- Address (MAR) is placed on address bus
- Control unit issues READ command
- Result (data from memory) appears on data bus
- Data from data bus copied into MDR (Co2),s03
- PC incremented by 1 (in parallel with data fetch from memory) (c03),s03
- Data (instruction) moved from MDR to IR (c04), S04
- MDR is now free for further data fetches (F 0 ; $\leftarrow$ E 1) ($\leftarrow$ c05) , s05

Sequence counter method

- Step 1: Identify the distinct phases in the flowchart. Employ log p number of flip flops to handle p number distinct phases

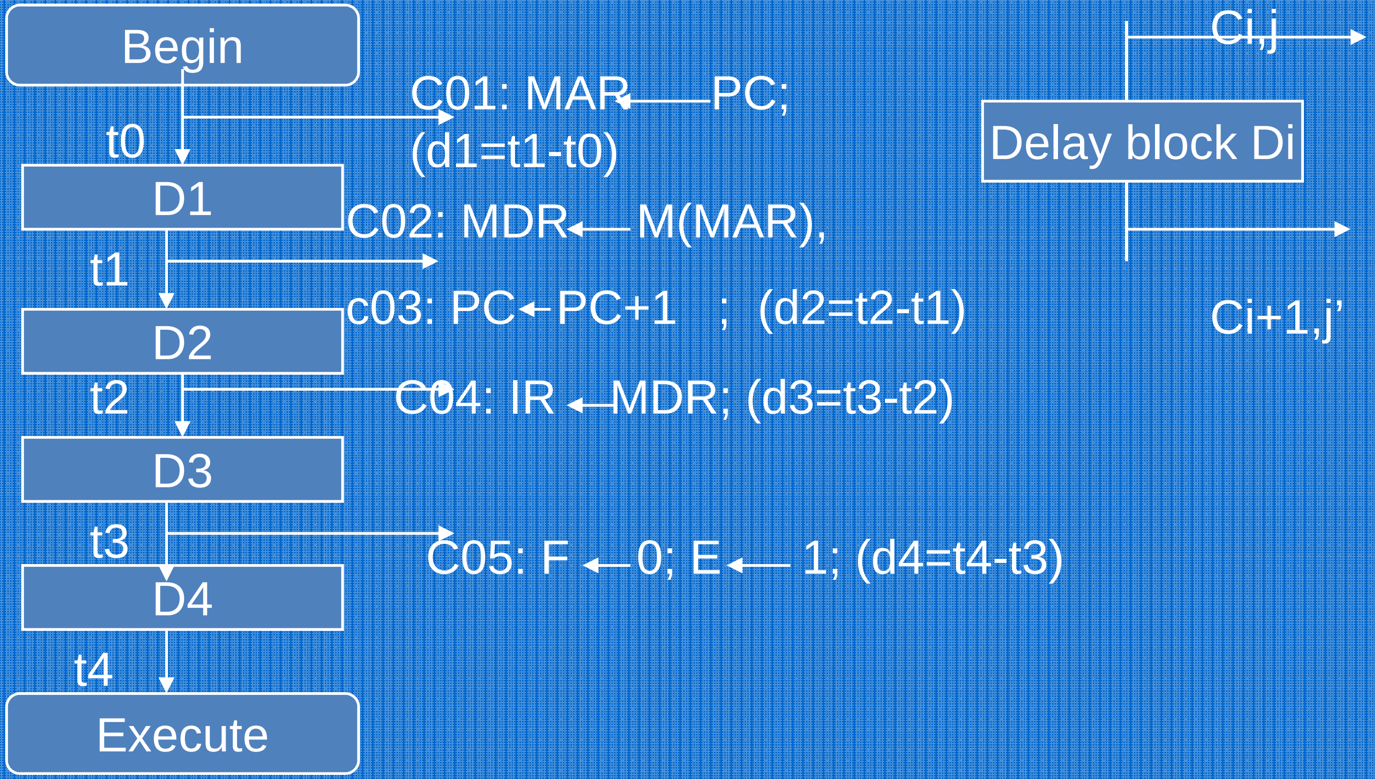


Step 2: Identify the maximum number of distinct steps, k, in each of the phases .Employ a mod k counter to generate control signals for each of the k steps

Step 3 : design a combinational logic circuit to generate the sequence of control signals to control the micro operations of each phase

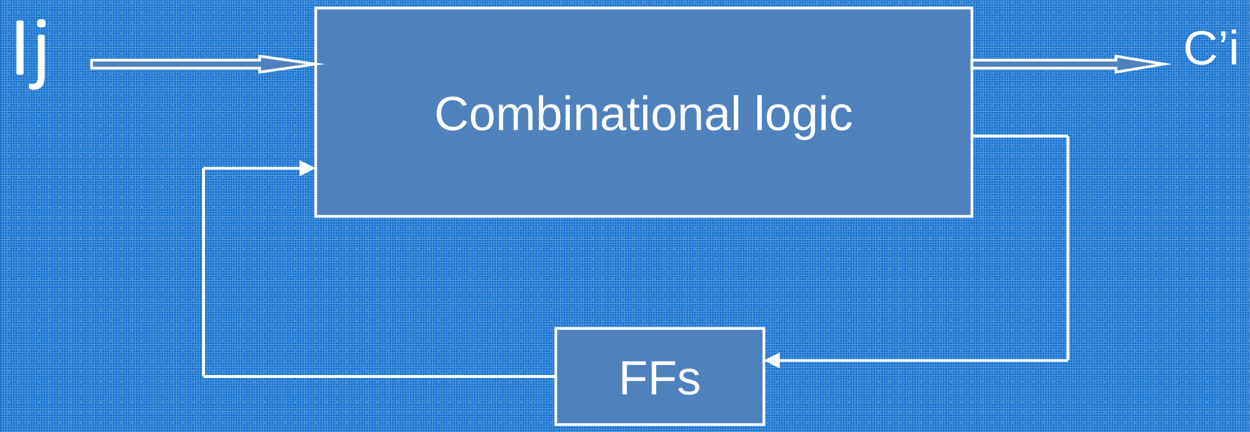
Delay element method

Control unit based on delay element method for the fetch cycle



State table method

- The state Si (i=1,2...) has been marked above each block of the flowchart.
- This the state of the controller which generates the control signals to control the micro operations in the data path



Steps to design the Control structure realizing flowchart

- State Assignment: States are assigned as s1,s2,s3…;each such assignment specifies a particular state of the controller at the specific time step. State table derived from state assignment
- State Minimization: A set of states {Sa,Sb,…Sc} can be merged to a single state S’ if Si & Sj is pair wise compatible
- State Encoding: State variables are defined and states are encoded in terms of state variables

Hard-wired Control Unit- advantages

1. Minimizes the average number of clock cycles needed per instruction
2. occupies a relatively small area (typically 10%) of the CPU chip area
3. High efficiency in terms of operation speed
4. is to minimize cost of the circuit

Problems With Hard Wired Designs

- Complex sequencing & micro-operation logic
- Difficult to design and test
- Inflexible design
- Large design turn around time for complex design
- Difficult to add new instructions

Tasks Done By Micro programmed Control Unit

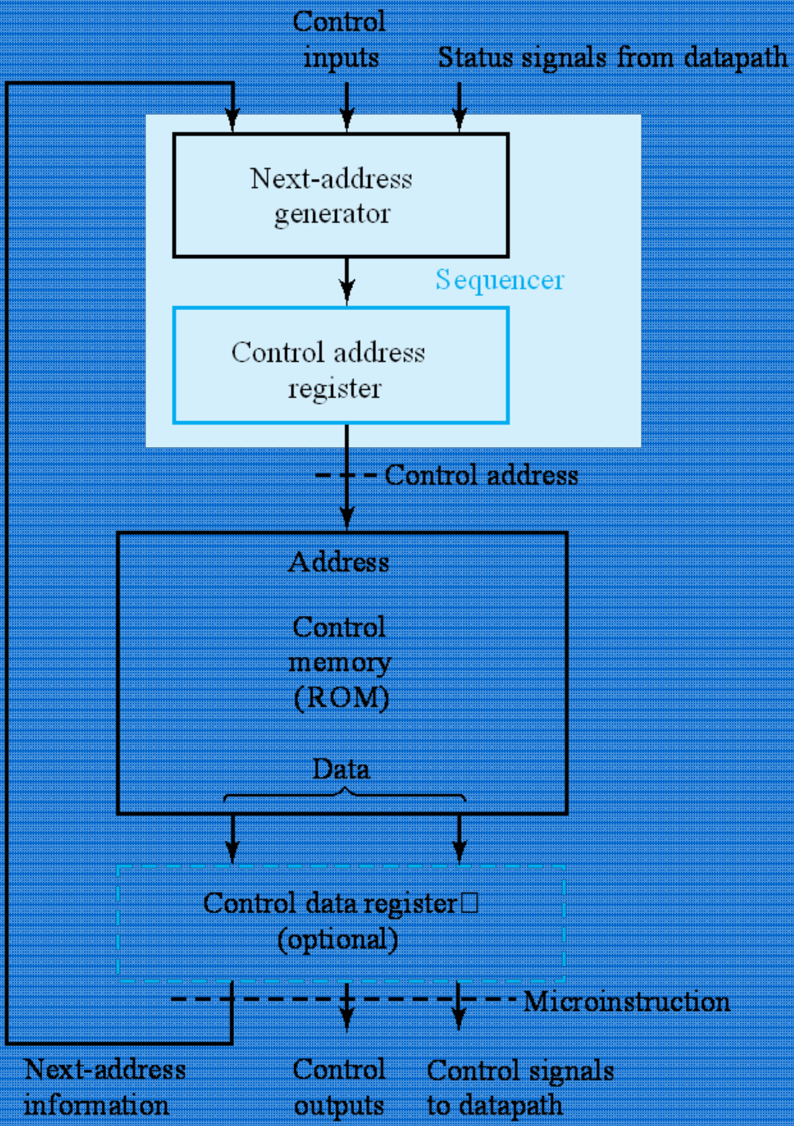
- Microinstruction sequencing
- Microinstruction execution
- Must consider both together

Microprogrammed Control

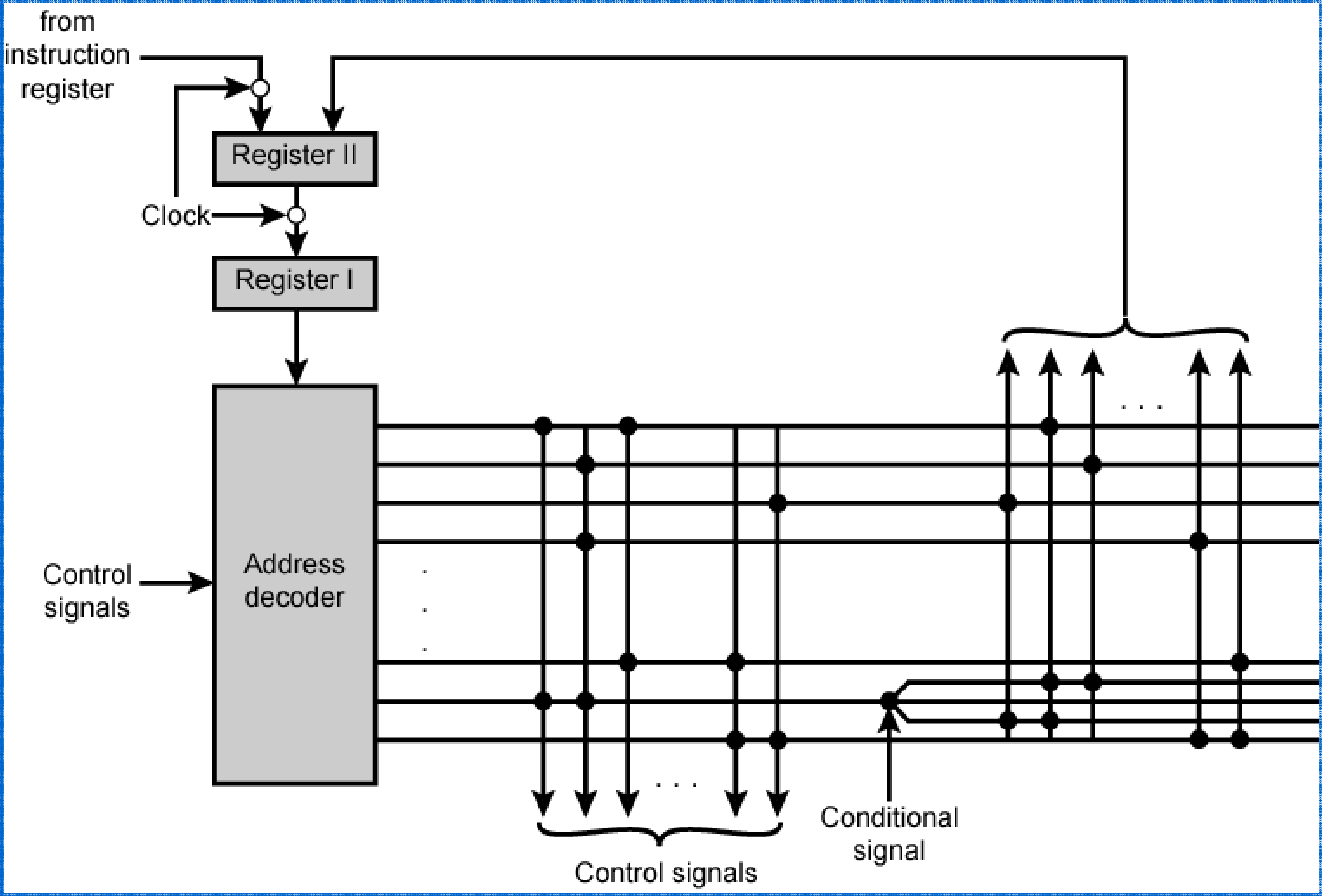
The concept of micro programmed control, employ the following steps:

1. Any instruction to be executed on a CPU can be broken down into a set of sequential micro operations – each specifying a RTL operation on the data path. The set of micro operations to be executed on the RTL components at any time step is referred as *microinstructions*.
2. The sequence of control signals necessary to execute the sequential microinstructions stored in ROM called *control ROM*
3. To implement an instruction on the data path , the control signals stored in the ROM can be accessed
4. The control signals read from the ROM are used to control the micro operations associated with a microinstruction to be executed at any time step
5. The address of the next micro instruction is generated
6. The steps 3,4 and 5 are repeated till the set of sequential microinstructions associated with the instruction is executed

Microprogrammed Control **(continued)**



Wilkes's Microprogrammed Control Unit



MICROINSTRUCTION FORMAT

Information in a Microinstruction

- **Control Information**
- **Sequencing Information**
- **Constant**

  Information which is useful when feeding into the system

These information needs to be organized in some way for

- **Efficient use of the microinstruction bits**
- **Fast decoding**

Field Encoding

- **Encoding the microinstruction bits**
- **Encoding slows down the execution speed due to the decoding delay**
- **Encoding also reduces the flexibility due to the decoding hardware**

Encoding of control signals

- Different formats of microinstruction depending on the encoding of control signals.
- The signals are divided into multiple control fields in a ROM word.

1. Horizontal/vertical formats
2. Functional encoding
3. Resource encoding
4. Direct versus indirect encoding

HORIZONTAL AND VERTICAL MICROINSTRUCTION FORMAT

Horizontal Microinstructions

Each bit directly controls each micro-operation or each control point

Horizontal **implies a long microinstruction word**

Advantages: Can control a variety of components operating in parallel.

--> Advantage of efficient hardware utilization

Disadvantages: Control word bits are not fully utilized

--> CS becomes large --> Costly

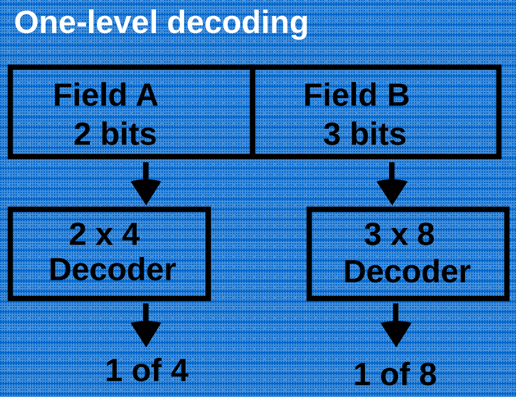
Vertical Microinstructions

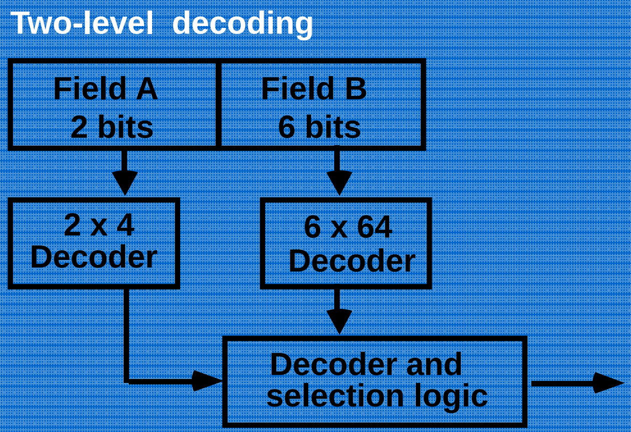
A microinstruction format that is not horizontal

Vertical **implies a short microinstruction word**

Encoded Microinstruction fields

--> Needs decoding circuits for one or two levels of decoding





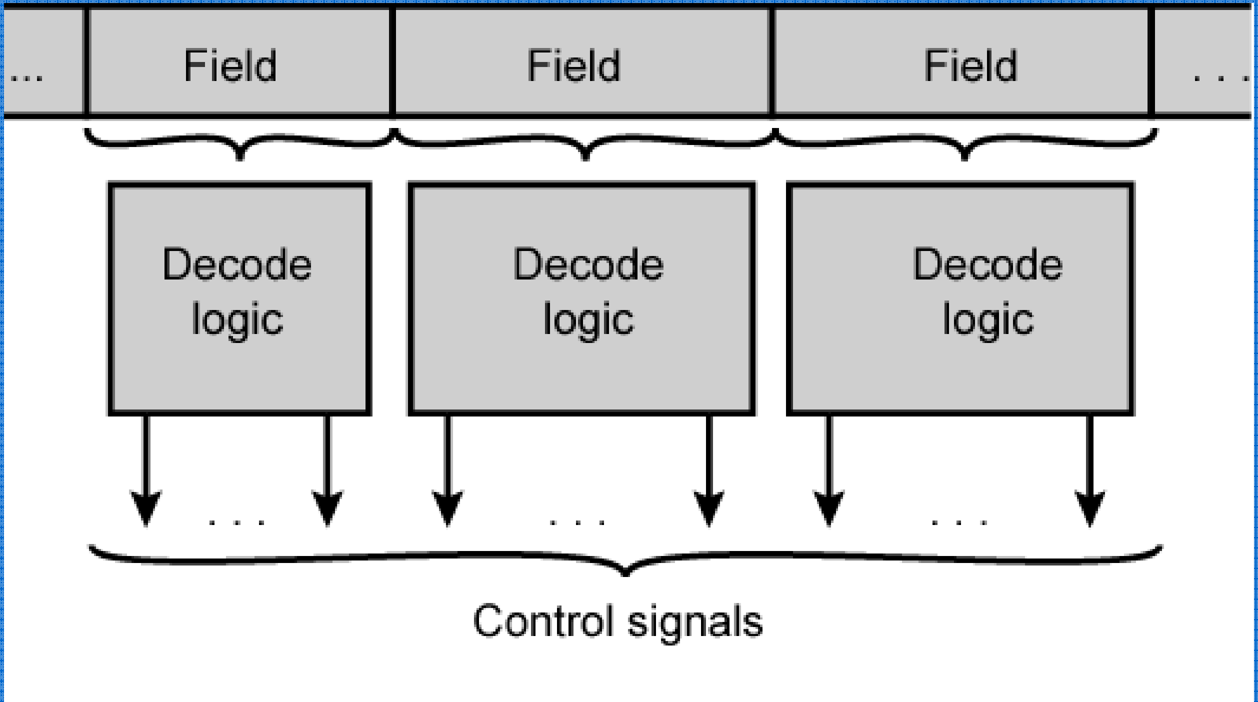
Functional Encoding

- The micro operations executed in a data path of a CPU may be categorized under different function names such as Shift function, Add function, Logical functions, Input-Output, etc
- Multiple control bits associated to control a specified function

Resource encoding

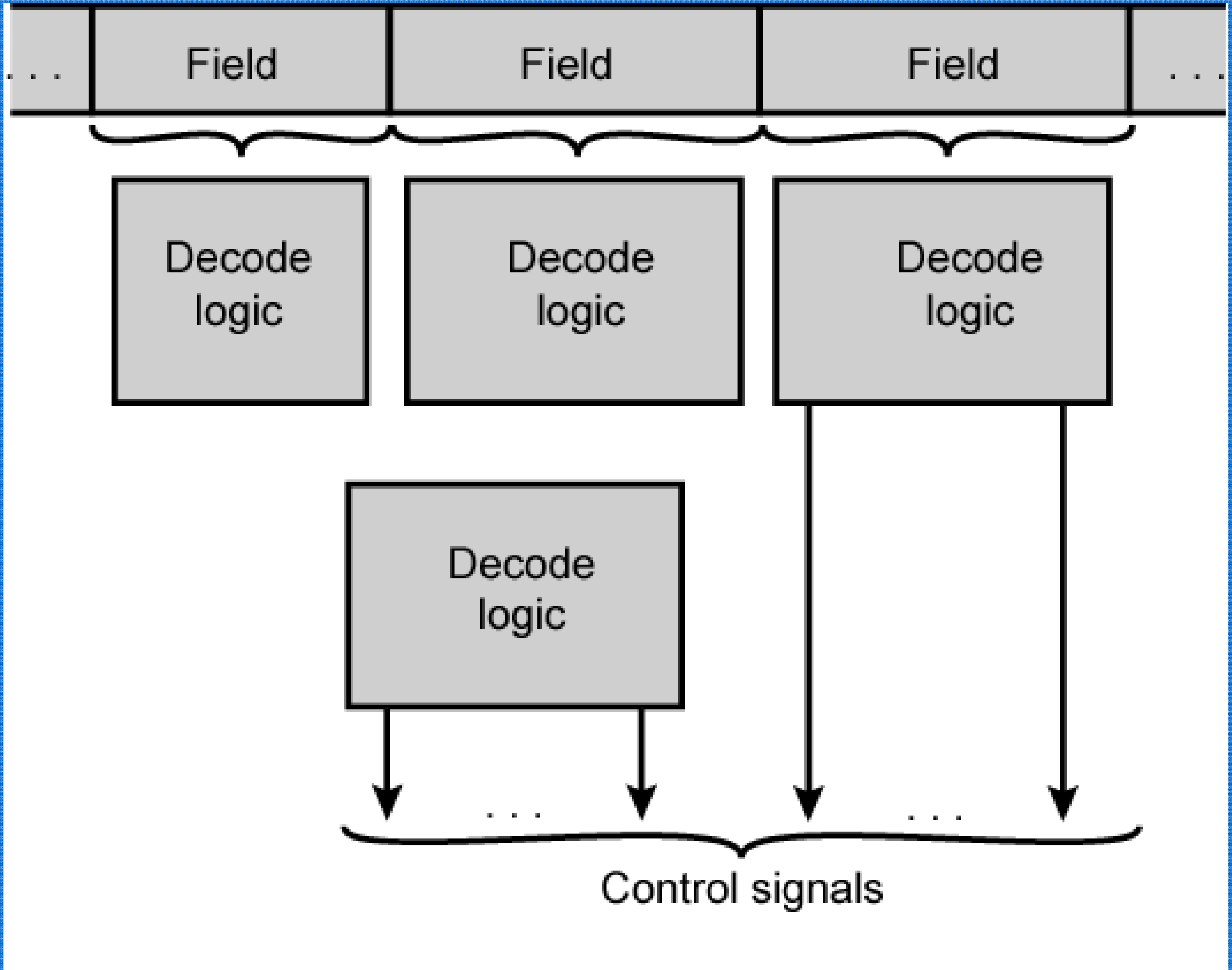
- The CPU data path consists of a set of interconnected RTL components.
- Each of these components or a subset of such components viewed as a resource.
- If control signals associated with such a resource are mutually exclusive, then they can be encoded in a single control field.

Microinstruction Encoding
Direct Encoding



(a) Direct encoding

Microinstruction Encoding
Indirect Encoding



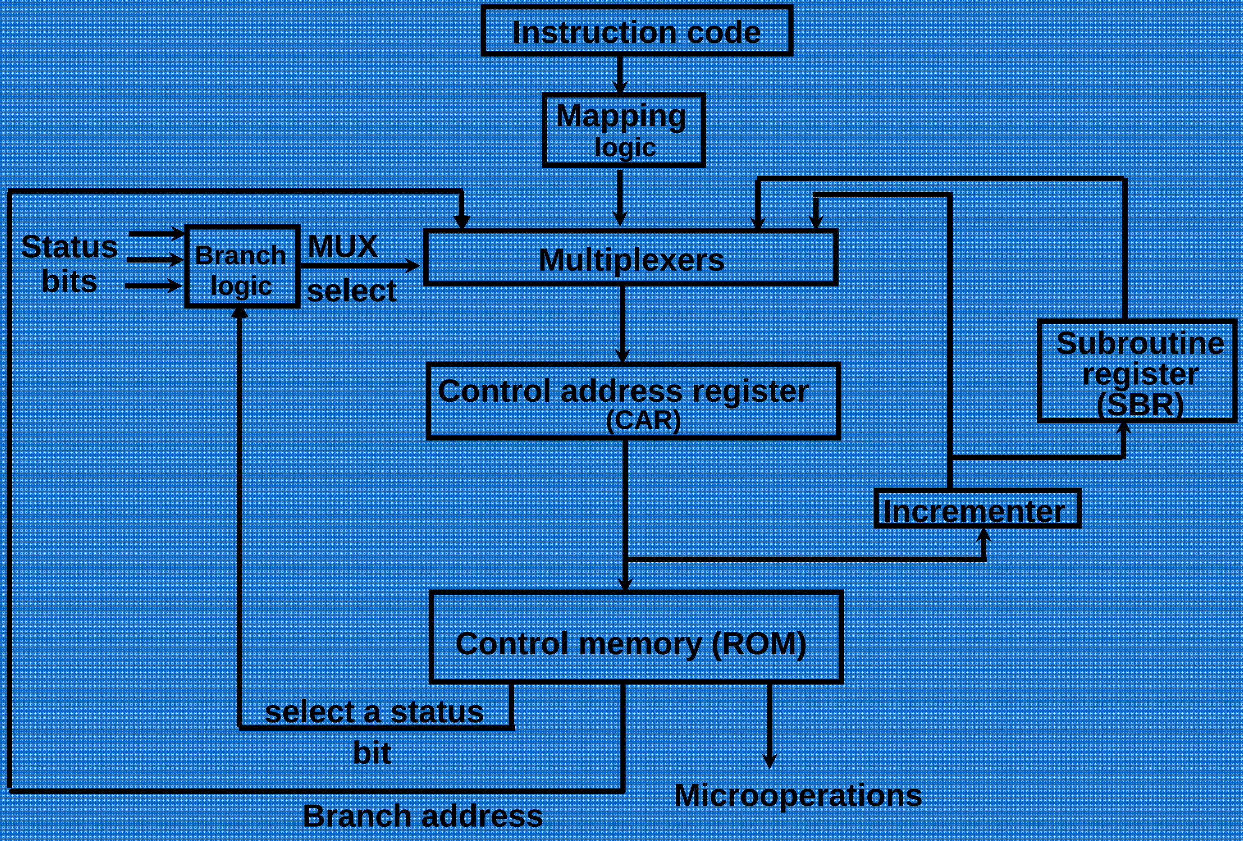
(b) Indirect encoding

Micro instruction execution

Two basic actions

1. Generating next microinstruction address and fetching micro program instruction from control memory by this address
2. Executing the micro operations controlled by different signals encoded in various control fields of the microinstruction
    1. Decoding and applying control signals on the CPU data path
    2. Executing the intended micro operations controlled by the signals
    3. Storing the output in the destination register specified in the micro operation on the CPU data path

MICROINSTRUCTION SEQUENCING



Sequencing Capabilities Required in a Control Storage

- Incrementing of the control address register
- Unconditional and conditional branches
- A mapping process from the bits of the machine instruction to an address for control memory
- A facility for subroutine call and return

Mono phase and Poly phase operation

- In a micro operation cycle, a micro program control unit implements the sequence of actions under 4 different phases.
- If the micro instruction cycle is controlled by a single clock pulse which synchronizes all the control signals ,then the mode is termed as *mono phase operation*
- If each of the phases is controlled by a different phase of a clock ,the mode is referred as poly phase operation

NANOSTORAGE AND NANOINSTRUCTION

The decoder circuits in a vertical micro program storage organization can be replaced by a ROM

=> Two levels of control storage

First level - *Control Storage*

Second level - *Nano Storage*

Two-level micro program

First level

-*Vertical* format Micro program

Second level

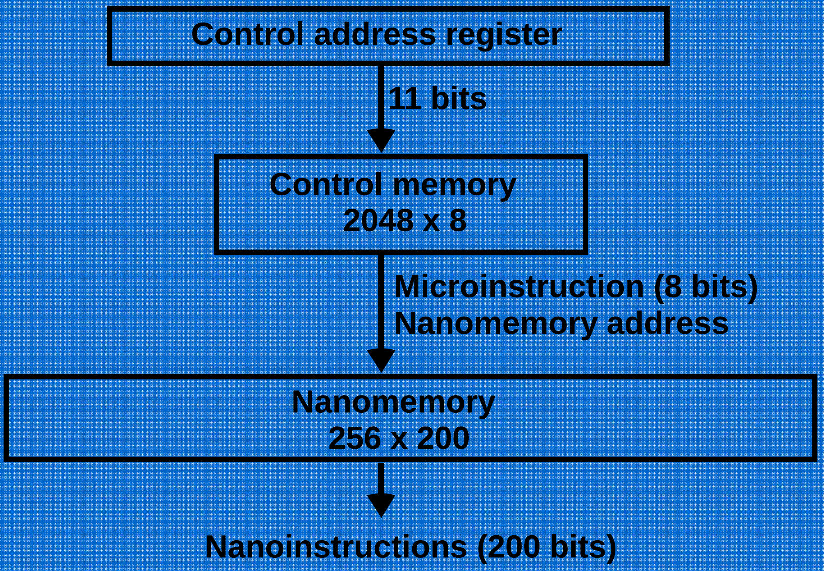
-*Horizontal* format Nano program

- Interprets the microinstruction fields, thus converts a vertical microinstruction format into a horizontal nano instruction format.

Usually, the micro program consists of a large number of short microinstructions, while the nano program contains fewer words with longer nano instructions.

TWO-LEVEL MICROPROGRAMMING - EXAMPLE

* Microprogram: 2048 microinstructions of 200 bits each
* With 1-Level Control Storage: 2048 x 200 = 409,600 bits
* Assumption:
  256 distinct microinstructions among 2048
* With 2-Level Control Storage:
  Nano Storage: 256 x 200 bits to store 256 distinct nanoinstructions
  Control storage: 2048 x 8 bits
  To address 256 nano storage locations 8 bits are needed
* Total 1-Level control storage: 409,600 bits
  Total 2-Level control storage: 67,584 bits (256 x 200 + 2048 x 8)



SRM UNIVERSITY (Under section 3 of UGC Act 1956)

Micro-programmed Control Unit - Advantages

1. A micro-programmed control unit is flexible and allows designers to incorporate new and more powerful instructions as VLSI technology increases the available chip area for the CPU
2. allows any design errors discovered during the prototyping stage to be removed

Microprogrammed Control Unit - Disadvantages

3. requires several clock cycles to execute each instruction, due to the access time of the microprogram memory
4. Occupies a large portion (typically 55%) of the CPU chip area