UNIT – IV

Microprogrammed Control: Control memory, Address sequencing, micro program example, design of control unit.

Central Processing Unit: General Register Organization, Instruction Formats, Addressing modes, Data Transfer and Manipulation, Program Control.

Computer Arithmetic: Addition and subtraction, multiplication Algorithms, Division Algorithms, Floating – point Arithmetic operations. Decimal Arithmetic unit, Decimal Arithmetic operations.

Microprogrammed Control

Generating control signals through stored microinstructions

Microprogrammed control uses a sequence of microinstructions stored in a special control memory.

Each microinstruction specifies control signals required to perform one or more microoperations.

The control unit fetches microinstructions and sequences them according to the current instruction and control conditions.

Compared with hardwired control, microprogramming can make complex instruction sets easier to implement and modify.

Basic elements include control memory, control address register, microinstruction register, sequencing logic, and control signal generation.

Control Memory

Stores microprograms / control words.

Sequencer

Selects the address of the next microinstruction.

Control Signals

Drive registers, ALU, buses and memory.

Control Memory

Purpose

A control memory stores the microinstructions that implement the control sequences of the processor.

It may be implemented using ROM or writable control storage.

Control Address Register

Holds the address of the current or next microinstruction.

The sequencing logic determines the next address.

Microinstruction Register

Holds the fetched microinstruction while its control fields are decoded or directly used to generate signals.

Control memory converts an instruction-level operation into a detailed sequence of internal register transfers and ALU operations.

Address Sequencing

Sequential

Next address = current address + 1. Used when microinstructions execute in normal sequence.

Branch

Next address is obtained from a specified address field or branch target.

Conditional Branch

Next address depends on a status condition such as a flag or external control signal.

Mapping

Instruction opcode can be mapped to the starting address of its microprogram.

Address sequencing provides the mechanism for selecting the next microinstruction and supports branches, subroutines, and instruction mapping.

Microprogram Example

Illustrative fetch and execution sequence

Fetch Sequence

T0: AR ← PC
T1: IR ← M[AR], PC ← PC + 1
T2: Decode IR

These operations prepare the processor for execution.

ADD Example

Fetch instruction → Decode → Obtain effective address → Read operand → AC ← AC + DR

Each step can be represented by a microinstruction.

Microinstruction Flow

Start → Fetch → Decode → Execute → Next instruction

A branch or mapping operation selects the appropriate execution sequence.

A microprogram is therefore a low-level control program that realizes the instruction set through sequences of elementary control actions.

Design of the Control Unit

Microinstruction Format

Control fields may specify:
- Register transfers
- ALU operation
- Memory/I/O actions
- Branch / sequencing information

Horizontal Microprogramming

Wide control word with many direct control bits.

Advantages: high parallelism and fast signal generation.
Trade-off: larger control memory.

Vertical Microprogramming

Encoded control fields reduce word width.

Advantages: smaller control memory.
Trade-off: additional decoding and less parallelism.

Design steps: identify microoperations → define control signals → create microinstruction format → design sequencing → store and verify the microprogram.

Central Processing Unit (CPU)

The CPU is the main processing component responsible for fetching, decoding, and executing instructions.

Major components are the Arithmetic Logic Unit (ALU), control unit, registers, and internal data paths.

The ALU performs arithmetic and logical operations on operands.

Registers provide fast storage for instructions, addresses, operands, results, and status information.

The control unit coordinates the activities of all CPU components using timing and control signals.

ALU

Arithmetic and logical processing

Registers

Fast internal storage

Control Unit

Coordinates instruction execution

General Register Organization

Register Set

Multiple general-purpose registers hold operands and intermediate results.

Example: R0, R1, R2, …, Rn.

Common Bus / MUX

Multiplexers select source registers and route data to the ALU or destination path.

ALU & Destination

The ALU performs the selected operation; the result is loaded into a selected destination register.

Control signals select source registers, ALU function, destination register, and status operations.

Instruction Formats

Three-Address

ADD R1, R2, R3
R1 ← R2 + R3

More operand fields; fewer instructions may be needed.

Two-Address

ADD R1, R2
R1 ← R1 + R2

One operand often serves as destination.

One-Address

ADD X
AC ← AC + M[X]

Accumulator is implied.

Zero-Address

Used in stack machines. Operands are implied by stack top.

Example: ADD

Instruction format determines how opcode, operands, registers, and addressing information are encoded in an instruction word.

Addressing Modes

Immediate

Operand is part of the instruction.

Direct

Address field gives the effective address.

Indirect

Address field points to a location containing the effective address.

Register

Operand is in a specified register.

Register Indirect

Register contains the effective address.

Indexed

Effective address is obtained using address + index register.

Relative

Effective address is based on PC plus displacement.

Auto Increment/ Decrement

Register is automatically updated before or after the access.

Data Transfer and Manipulation

Data Transfer

LOAD / MOVE
STORE
EXCHANGE

Move data between registers, memory and I/O

Arithmetic

ADD
SUBTRACT
INCREMENT
DECREMENT

Used for numeric computation.

Logical & Shift

AND, OR, XOR, NOT
Shift left/right
Rotate

Used for bit-level manipulation.

Data manipulation instructions transform operands while preserving or updating relevant status flags.

Program Control

Branch

Unconditional branch changes the PC to a target address.

Conditional branch changes control flow only when a specified condition is true.

Subroutine

CALL / BSA saves a return address and transfers control to a procedure.

RETURN restores control to the calling program.

Program Counter & Stack

PC identifies the next instruction.

A stack can store return addresses, saved registers, and temporary data during nested calls.

Program-control instructions allow decisions, loops, function calls, returns, and exception/interrupt handling.

Computer Arithmetic – Addition, Subtraction & Multiplication

Addition / Subtraction

Binary addition uses full adders.

Subtraction can use 2's complement:
A − B = A + (2's complement of B).

Shift-and-Add Multiplication

Examine multiplier bits.
If bit = 1, add shifted multiplicand.
Shift multiplicand and multiplier each step.
Accumulate the partial products.

Booth's Algorithm

Efficient signed multiplication using 2's complement.

Examines adjacent multiplier bits and performs add/subtract/shift operations, reducing repeated additions for runs of 1s.

Arithmetic algorithms trade hardware cost, execution time, precision, and complexity.

Division Algorithms & Floating-Point Arithmetic

Restoring Division

Shift partial remainder and quotient.
Subtract divisor.
If result is negative, restore the r emainder and set quotient bit to 0.

Non-Restoring Division

Avoids an explicit restore step after every negative partial remainder.
Uses add/subtract decisions in successive iterations.

Floating-Point Add/ Subtract

1. Compare exponents
2. Align significands
3. Add/subtract
4. Normalize
5. Round

FP Multiply/Divide

Combine signs.
Operate on significands.
Add/subtract exponents.
Normalize and round.

Floating-point arithmetic supports a large dynamic range but must manage precision, normalization, overflow, underflow, and rounding.

Decimal Arithmetic Unit & Unit–IV Summary

Decimal Arithmetic Unit

Processes decimal digits using BCD representation and correction logic.

BCD addition commonly adds 0110 when a digit result is greater than 9 or a carry is generated.

Decimal Operations

Decimal addition and subtraction can be implemented with BCD arithmetic circuits.

Decimal arithmetic is useful where exact decimal digit representation is important.

Unit Summary

Microprogrammed Control → Control Memory → Address Sequencing → CPU → Registers → Instruction Formats → Addressing Modes → Program Control → Arithmetic Algorithms → Decimal Arithmetic.

Key idea: The control unit coordinates processor operations, while the CPU datapath and arithmetic units execute data-processing instructions.