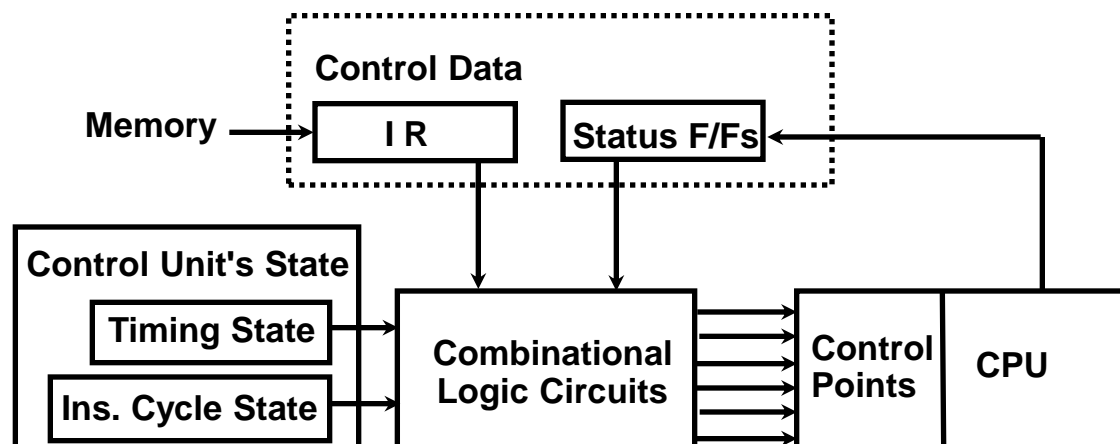# MICROPROGRAMMED  CONTROL

- **Control Memory**

- **Sequencing Microinstructions**

- **Microprogram Example**

- **Design of Control Unit**
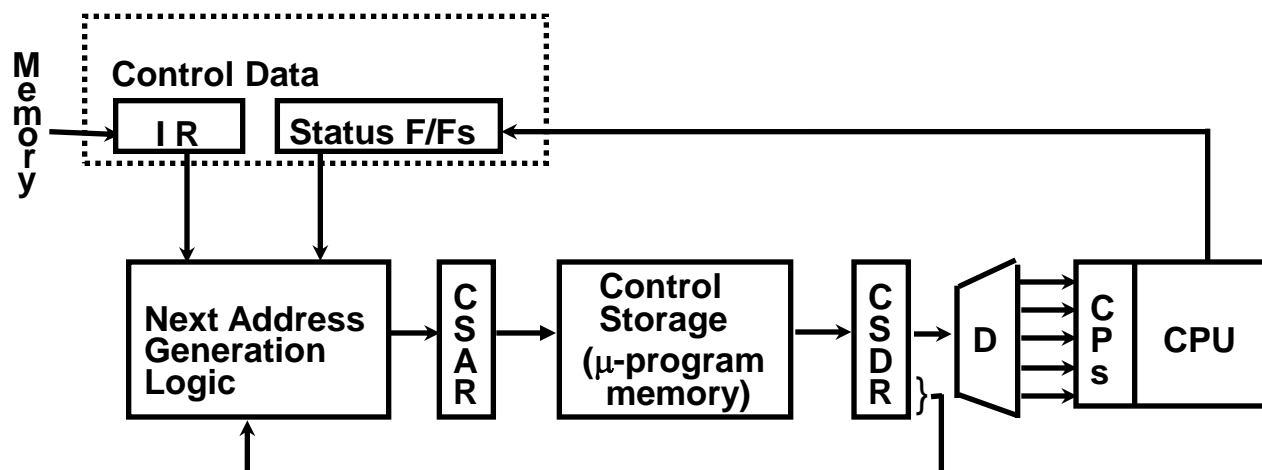
# COMPARISON OF CONTROL UNIT IMPLEMENTATIONS

## Control Unit Implementation

### Combinational Logic Circuits (Hard-wired)

```
                          Control Data
      Memory ──────▶ [    I R    ]      [ Status F/Fs ]◀────────┐
                                                                 │
   ┌─ Control Unit's State ─┐                                    │
   │  [  Timing State  ]────┼──▶  Combinational  ═══▶ Control   CPU
   │                        │      Logic Circuits ═══▶ Points
   │  [ Ins. Cycle State ]──┼──▶                  ═══▶
```

### Microprogram

```
   M    Control Data
   e  [   I R   ]  [ Status F/Fs ]◀─────────────────────────┐
   m                                                         │
   o                                                         │
   r                                                         │
   y        │         │                                      │
            ▼         ▼                                      │
      ┌──────────────┐  ┌─┐  ┌──────────┐  ┌─┐              │
      │ Next Address │  │C│  │ Control  │  │C│   ╱│   C      │
      │ Generation   │─▶│S│─▶│ Storage  │─▶│S│─▶│D│═▶P═▶ CPU │
      │ Logic        │  │A│  │(μ-program│  │D│   ╲│   s      │
      └──────────────┘  │R│  │ memory)  │  │R│}            │
            ▲           └─┘  └──────────┘  └─┘              │
            └─────────────────────────────────────────────┘
```

# 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
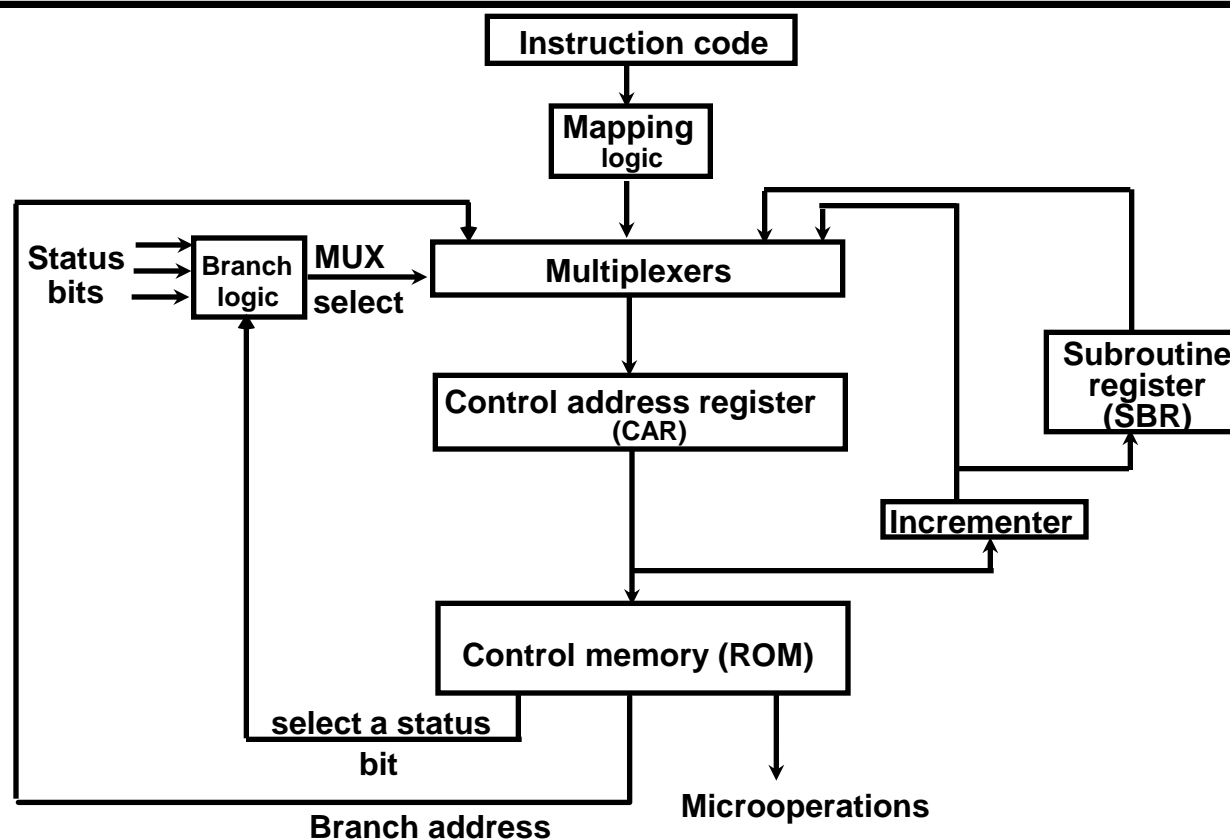  - Microprogram can be changed by a systems programmer or a user

# TERMINOLOGY

*Sequencer (Microprogram Sequencer)*

**A Microprogram 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**

# MICROINSTRUCTION  SEQUENCING

```
                        ┌─────────────────┐
                        │ Instruction code │
                        └─────────────────┘
                                 │
                        ┌─────────────────┐
                        │     Mapping      │
                        │      logic       │
                        └─────────────────┘
```

| | Branch logic | MUX | Multiplexers | | Subroutine register (SBR) |
|---|---|---|---|---|---|

**Status bits** → **Branch logic** → **MUX select** → **Multiplexers**

**Control address register (CAR)**

**Subroutine register (SBR)**

**Incrementer**

**Control memory (ROM)**

**select a status bit**

**Branch address**

**Microoperations**

## 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**

# MAPPING OF INSTRUCTIONS

**Direct Mapping**

**OP-codes of Instructions**

| | |
|---|---|
| ADD | 0000 |
| AND | 0001 |
| LDA | 0010 |
| STA | 0011 |
| BUN | 0100 |

**Mapping Bits**    10 XXXX 010

Address

| Address | |
|---|---|
| 0000 | ADD Routine |
| 0001 | AND Routine |
| 0010 | LDA Routine |
| 0011 | STA Routine |
| 0100 | BUN Routine |
| | Control Storage |

| Address | |
|---|---|
| 10 0000 010 | ADD Routine |
| 10 0001 010 | AND Routine |
| 10 0010 010 | LDA Routine |
| 10 0011 010 | STA Routine |
| 10 0100 010 | BUN Routine |

# MAPPING OF INSTRUCTIONS TO MICROROUTINES

**Mapping from the OP-code of an instruction to the address of the Microinstruction which is the starting microinstruction of its execution microprogram**
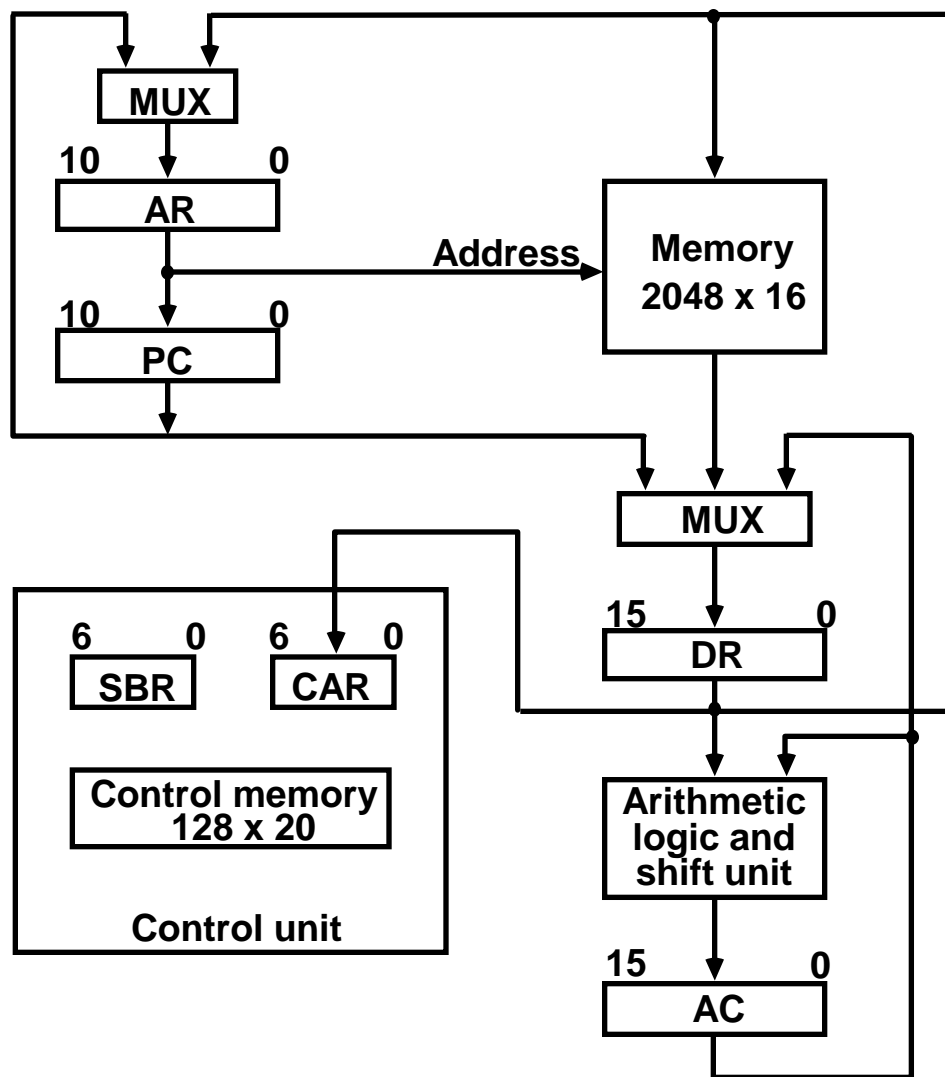
|  | OP-code | |
|---|---|---|
| Machine Instruction | 1 0 1 1 | Address |

| Mapping bits | 0 | x x x x | 0 0 |
|---|---|---|---|

| Microinstruction address | 0 1 0 1 1 0 0 |
|---|---|
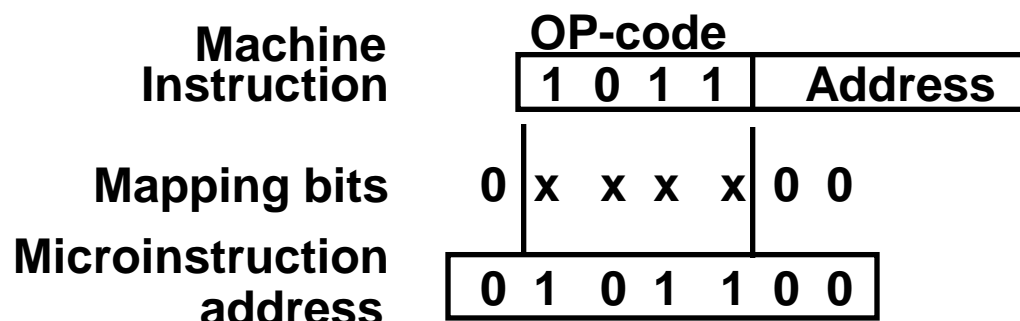
**Mapping function implemented by ROM or PLA**

```
          ┌──────────────┐
          │   OP-code    │
          └──────────────┘
                 │
                 ▼
      ┌─────────────────────┐
      │   Mapping memory    │
      │   (ROM or PLA)      │
      └─────────────────────┘
                 │
                 ▼
  ┌────────────────────────────┐
  │  Control address register  │
  └────────────────────────────┘
                 │
                 ▼
      ┌─────────────────────┐
      │   Control Memory    │
      └─────────────────────┘
```
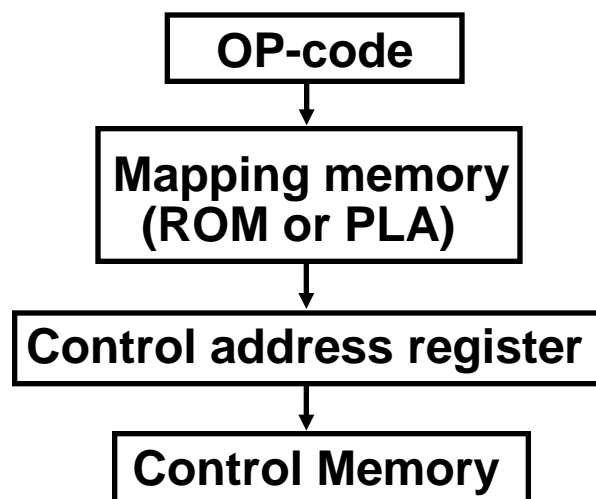
# MICROPROGRAM   EXAMPLE

**Computer Configuration**

# MACHINE INSTRUCTION FORMAT

## Machine instruction format

| 15 14 | 11 10 | 0 |
|---|---|---|
| I | Opcode | Address |
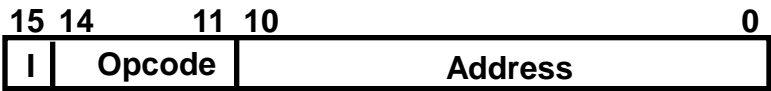
## Sample machine instructions

| Symbol | OP-code | Description |
|---|---|---|
| ADD | 0000 | AC ← AC + M[EA] |
| BRANCH | 0001 | if (AC < 0) then (PC ← EA) |
| STORE | 0010 | M[EA] ← AC |
| EXCHANGE | 0011 | AC ← M[EA], M[EA] ← AC |

**EA is the effective address**

## Microinstruction Format

| 3 | 3 | 3 | 2 | 2 | 7 |
|---|---|---|---|---|---|
| F1 | F2 | F3 | CD | BR | AD |
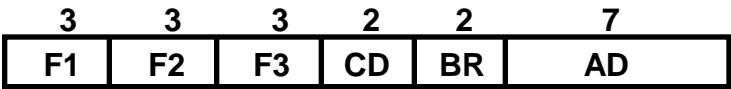
**F1, F2, F3: Microoperation fields**
**CD: Condition for branching**
**BR: Branch field**
**AD: Address field**

# MICROINSTRUCTION  FIELD  DESCRIPTIONS - F1,F2,F3

| F1 | Microoperation | Symbol |
|-----|----------------|--------|
| 000 | None | NOP |
| 001 | $AC \leftarrow AC + DR$ | ADD |
| 010 | $AC \leftarrow 0$ | CLRAC |
| 011 | $AC \leftarrow AC + 1$ | INCAC |
| 100 | $AC \leftarrow DR$ | DRTAC |
| 101 | $AR \leftarrow DR(0\text{-}10)$ | DRTAR |
| 110 | $AR \leftarrow PC$ | PCTAR |
| 111 | $M[AR] \leftarrow DR$ | WRITE |

| F2 | Microoperation | Symbol |
|-----|----------------|--------|
| 000 | None | NOP |
| 001 | $AC \leftarrow AC - DR$ | SUB |
| 010 | $AC \leftarrow AC \vee DR$ | OR |
| 011 | $AC \leftarrow AC \wedge DR$ | AND |
| 100 | $DR \leftarrow M[AR]$ | READ |
| 101 | $DR \leftarrow AC$ | ACTDR |
| 110 | $DR \leftarrow DR + 1$ | INCDR |
| 111 | $DR(0\text{-}10) \leftarrow PC$ | PCTDR |

| F3 | Microoperation | Symbol |
|-----|----------------|--------|
| 000 | None | NOP |
| 001 | $AC \leftarrow AC \oplus DR$ | XOR |
| 010 | $AC \leftarrow AC'$ | COM |
| 011 | $AC \leftarrow shl\ AC$ | SHL |
| 100 | $AC \leftarrow shr\ AC$ | SHR |
| 101 | $PC \leftarrow PC + 1$ | INCPC |
| 110 | $PC \leftarrow AR$ | ARTPC |
| 111 | Reserved | |

# MICROINSTRUCTION  FIELD  DESCRIPTIONS - CD, BR

| CD | Condition | Symbol | Comments |
|----|-----------|--------|----------|
| 00 | Always = 1 | U | Unconditional branch |
| 01 | DR(15) | I | Indirect address bit |
| 10 | AC(15) | S | Sign bit of AC |
| 11 | AC = 0 | Z | Zero value in AC |

| BR | Symbol | Function |
|----|--------|----------|
| 00 | JMP | CAR ← AD if condition = 1 |
| | | CAR ←  CAR + 1 if condition = 0 |
| 01 | CALL | CAR ← AD, SBR ← CAR + 1 if condition = 1 |
| | | CAR ← CAR + 1 if condition = 0 |
| 10 | RET | CAR ← SBR (Return from subroutine) |
| 11 | MAP | CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0 |

# SYMBOLIC  MICROINSTRUCTIONS

- **Symbols are used in microinstructions as in assembly language**
- **A symbolic microprogram can be translated into its binary equivalent by a microprogram assembler.**

**Sample Format**

**five fields:          label; micro-ops; CD; BR; AD**

**Label:               may be empty or may specify a symbolic address terminated with a colon**

**Micro-ops: consists of one, two, or three symbols separated by commas**

**CD:      one of {U, I, S, Z}, where        U: Unconditional Branch**
**                                            I:  Indirect address bit**
**                                            S: Sign of AC**
**                                            Z:  Zero value in AC**

**BR:      one of {JMP, CALL, RET, MAP}**

**AD:      one of {Symbolic address, NEXT, empty}**

# SYMBOLIC MICROPROGRAM - FETCH ROUTINE

**During FETCH, Read an instruction from memory and decode the instruction and update PC**

## Sequence of microoperations in the fetch cycle:

```
AR ←  PC
DR ←  M[AR], PC ← PC + 1
AR ← DR(0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0
```

## Symbolic microprogram for the fetch cycle:

```
            ORG 64
FETCH:      PCTAR          U   JMP   NEXT
            READ, INCPC    U   JMP   NEXT
            DRTAR          U   MAP
```

## Binary equivalents translated by an assembler

| Binary address | F1 | F2 | F3 | CD | BR | AD |
|---|---|---|---|---|---|---|
| 1000000 | 110 | 000 | 000 | 00 | 00 | 1000001 |
| 1000001 | 000 | 100 | 101 | 00 | 00 | 1000010 |
| 1000010 | 101 | 000 | 000 | 00 | 11 | 0000000 |

# SYMBOLIC  MICROPROGRAM

- **Control Storage:  128  20-bit words**
- **The first 64 words: Routines for the 16 machine instructions**
- **The last 64 words:  Used for other purpose (e.g., fetch routine and other subroutines)**
- **Mapping:**          **OP-code XXXX into 0XXXX00, the first address for the 16 routines are**
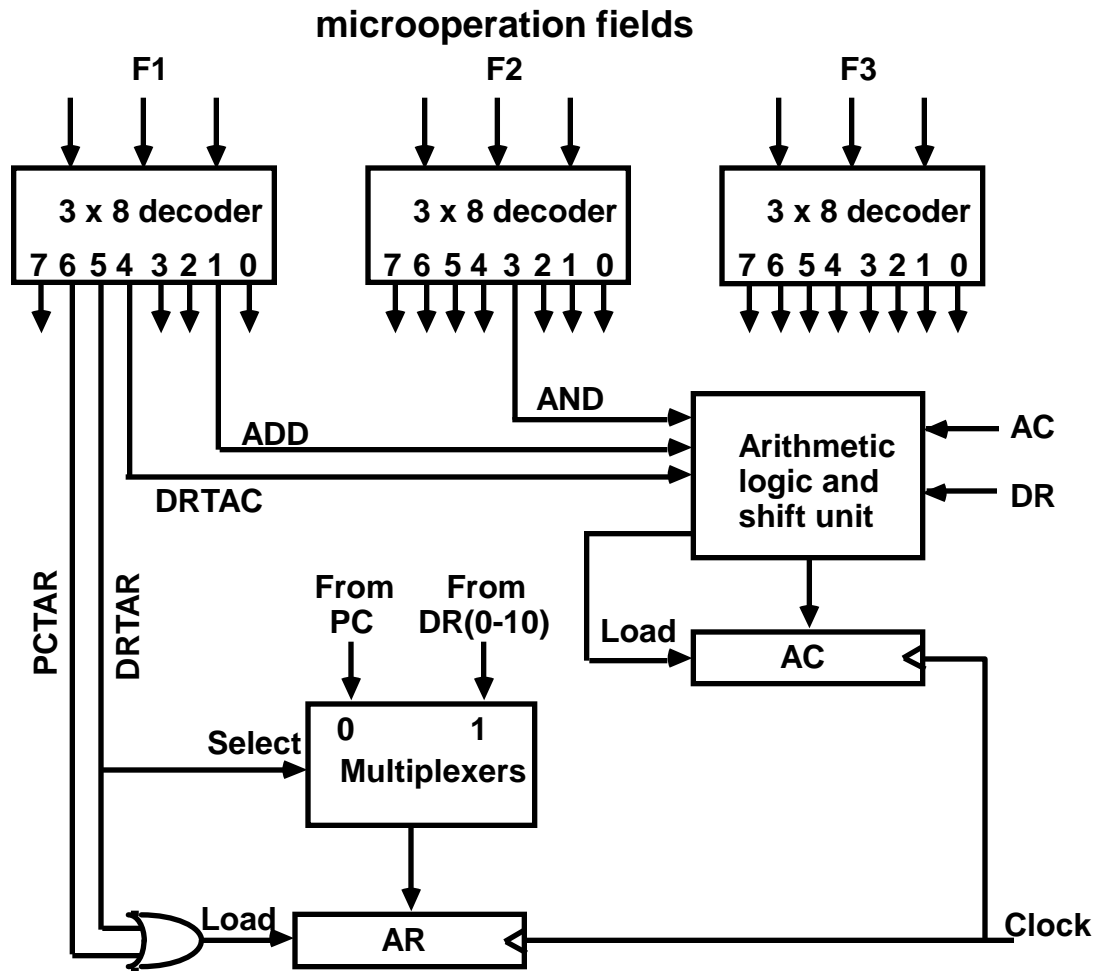  **0(0 0000 00), 4(0 0001 00),  8, 12, 16, 20, ..., 60**

## Partial Symbolic Microprogram

| Label | Microops | CD | BR | AD |
|---|---|---|---|---|
| | ORG 0 | | | |
| ADD: | NOP | I | CALL | INDRCT |
| | READ | U | JMP | NEXT |
| | ADD | U | JMP | FETCH |
| | | | | |
| | ORG 4 | | | |
| BRANCH: | NOP | S | JMP | OVER |
| | NOP | U | JMP | FETCH |
| OVER: | NOP | I | CALL | INDRCT |
| | ARTPC | U | JMP | FETCH |
| | | | | |
| | ORG 8 | | | |
| STORE: | NOP | I | CALL | INDRCT |
| | ACTDR | U | JMP | NEXT |
| | WRITE | U | JMP | FETCH |
| | | | | |
| | ORG 12 | | | |
| EXCHANGE: | NOP | I | CALL | INDRCT |
| | READ | U | JMP | NEXT |
| | ACTDR, DRTAC | U | JMP | NEXT |
| | WRITE | U | JMP | FETCH |
| | | | | |
| | ORG 64 | | | |
| FETCH: | PCTAR | U | JMP | NEXT |
| | READ, INCPC | U | JMP | NEXT |
| | DRTAR | U | MAP | |
| INDRCT: | READ | U | JMP | NEXT |
| | DRTAR | U | RET | |

# BINARY MICROPROGRAM

| Micro Routine | Address | | Binary Microinstruction | | | | | |
| | Decimal | Binary | F1 | F2 | F3 | CD | BR | AD |
|---|---|---|---|---|---|---|---|---|
| ADD | 0 | 0000000 | 000 | 000 | 000 | 01 | 01 | 1000011 |
| | 1 | 0000001 | 000 | 100 | 000 | 00 | 00 | 0000010 |
| | 2 | 0000010 | 001 | 000 | 000 | 00 | 00 | 1000000 |
| | 3 | 0000011 | 000 | 000 | 000 | 00 | 00 | 1000000 |
| BRANCH | 4 | 0000100 | 000 | 000 | 000 | 10 | 00 | 0000110 |
| | 5 | 0000101 | 000 | 000 | 000 | 00 | 00 | 1000000 |
| | 6 | 0000110 | 000 | 000 | 000 | 01 | 01 | 1000011 |
| | 7 | 0000111 | 000 | 000 | 110 | 00 | 00 | 1000000 |
| STORE | 8 | 0001000 | 000 | 000 | 000 | 01 | 01 | 1000011 |
| | 9 | 0001001 | 000 | 101 | 000 | 00 | 00 | 0001010 |
| | 10 | 0001010 | 111 | 000 | 000 | 00 | 00 | 1000000 |
| | 11 | 0001011 | 000 | 000 | 000 | 00 | 00 | 1000000 |
| EXCHANGE | 12 | 0001100 | 000 | 000 | 000 | 01 | 01 | 1000011 |
| | 13 | 0001101 | 001 | 000 | 000 | 00 | 00 | 0001110 |
| | 14 | 0001110 | 100 | 101 | 000 | 00 | 00 | 0001111 |
| | 15 | 0001111 | 111 | 000 | 000 | 00 | 00 | 1000000 |
| FETCH | 64 | 1000000 | 110 | 000 | 000 | 00 | 00 | 1000001 |
| | 65 | 1000001 | 000 | 100 | 101 | 00 | 00 | 1000010 |
| | 66 | 1000010 | 101 | 000 | 000 | 00 | 11 | 0000000 |
| INDRCT | 67 | 1000011 | 000 | 100 | 000 | 00 | 00 | 1000100 |
| | 68 | 1000100 | 101 | 000 | 000 | 00 | 10 | 0000000 |

**This microprogram can be implemented using ROM**

# DESIGN OF CONTROL UNIT
## - DECODING ALU CONTROL INFORMATION -

**microoperation fields**

# MICROPROGRAM SEQUENCER
## - NEXT MICROINSTRUCTION ADDRESS LOGIC -



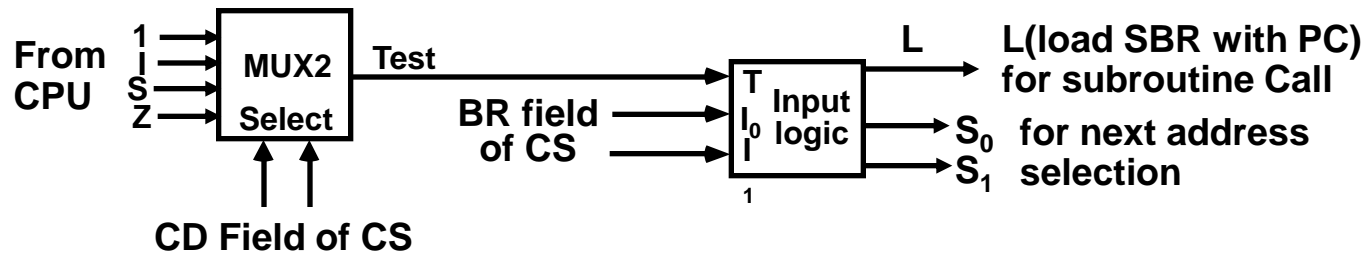| $S_1 S_0$ | Address Source |
|-----------|----------------|
| 00 | CAR + 1, In-Line |
| 01 | SBR RETURN |
| 10 | CS(AD), Branch or CALL |
| 11 | MAP |
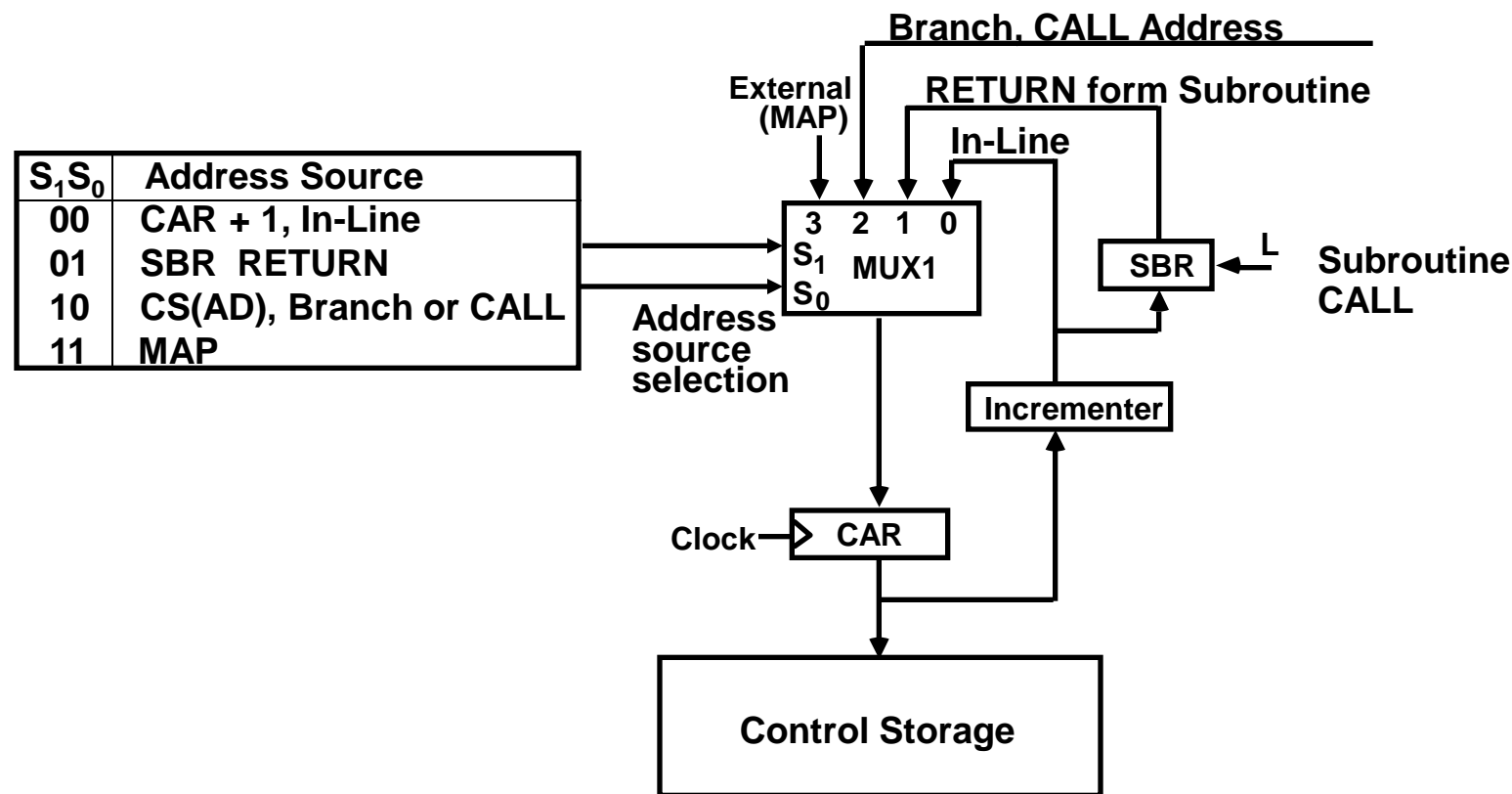
**MUX-1 selects an address from one of four sources and routes it into a CAR**

- In-Line Sequencing $\rightarrow$ CAR + 1
- Branch, Subroutine Call $\rightarrow$ CS(AD)
- Return from Subroutine $\rightarrow$ Output of SBR
- New Machine instruction $\rightarrow$ MAP

# MICROPROGRAM SEQUENCER
## - CONDITION AND BRANCH CONTROL -



From CPU → 1, I, S, Z → MUX2 Select → Test

CD Field of CS

BR field of CS → T, $I_0$, $I_1$ → Input logic → L, $S_0$, $S_1$

L(load SBR with PC) for subroutine Call
$S_0$ for next address
$S_1$ selection

## Input Logic

| $I_0 I_1 T$ | Meaning | Source of Address | $S_1 S_0$ | L |
|---|---|---|---|---|
| 000 | In-Line | CAR+1 | 00 | 0 |
| 001 | JMP | CS(AD) | 10 | 0 |
| 010 | In-Line | CAR+1 | 00 | 0 |
| 011 | CALL | CS(AD) and SBR <- CAR+1 | 10 | 1 |
| 10x | RET | SBR | 01 | 0 |
| 11x | MAP | DR(11-14) | 11 | 0 |

$$S_0 = I_0$$
$$S_1 = I_0 I_1 + I_0'T$$
$$L = I_0'I_1 T$$

# MICROPROGRAM SEQUENCER