Microprogrammed Control

Control Unit

- Major functional parts of digital computer – CPU, Memory and Input – Output.
- Main digital hardware functional units of CPU are Control unit, ALU and registers.
- The function of control unit is to initiate sequences of micro operations.
- The numbers of available micro operations in a given system is finite.

Control Unit

- When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be **hardwired control unit**.

- Microprogramming is a second alternative for designing the control unit of a digital computer.

- A control unit whose binary control variables are stored in memory is called a **microprogrammed control unit**.

Control Unit

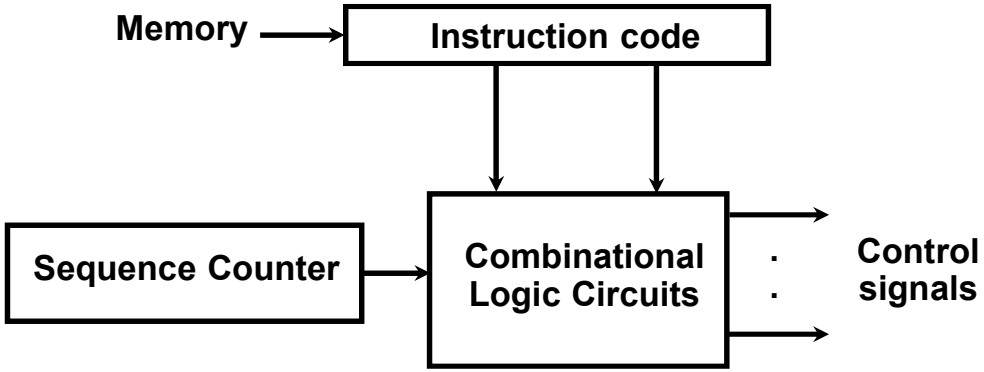
- The control function that specifies a micro operation is a binary variable. When it is in ‘one’ binary state, the corresponding micro operation is executed.
- The ‘Active’ state of a control variable may be ‘one’ may be ‘zero’.

Control Unit

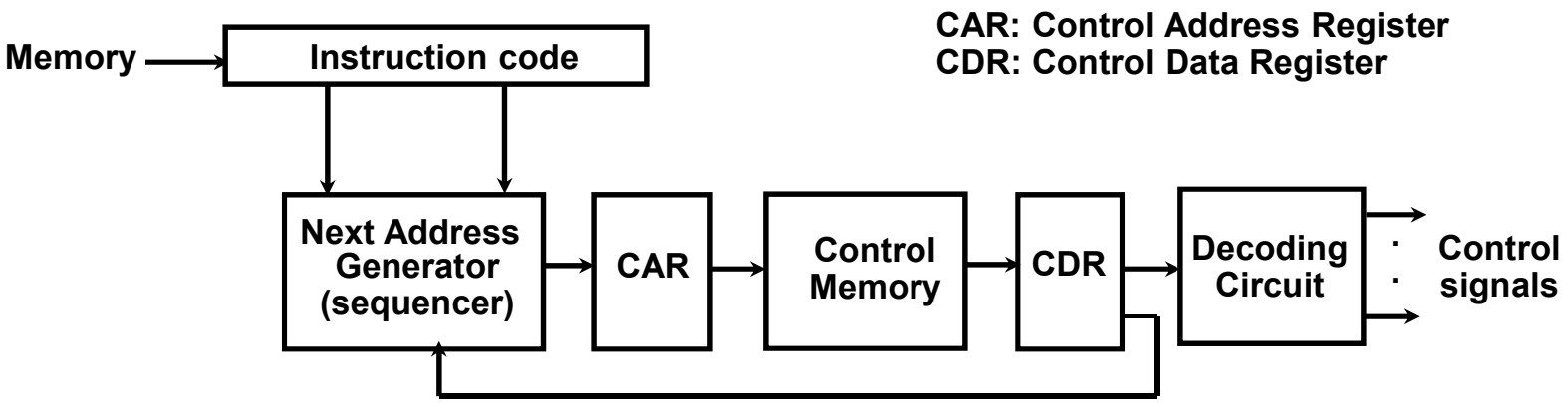
- The control unit initiates a series of sequential steps of micro operations.
- During any given time, certain micro operations are to be initiated while others remain idle.
- The control variables at any given time can be represented by a string of 1's and 0's called a control word.
- As such, control words can be programmed to perform various operations on the components of the system.
- Each word in control memory contains a micro instruction.

Control Unit Implementation

- Hardwired



- Microprogrammed



Microprogrammed Control Unit

- Control signals
  - Group of bits used to select paths in multiplexers, decoders, arithmetic logic units
- Control variables
  - Binary variables specify microoperations
    - Certain microoperations initiated while others idle
- Control word
  - String of 1's and 0's represent control variables

Microprogrammed Control Unit

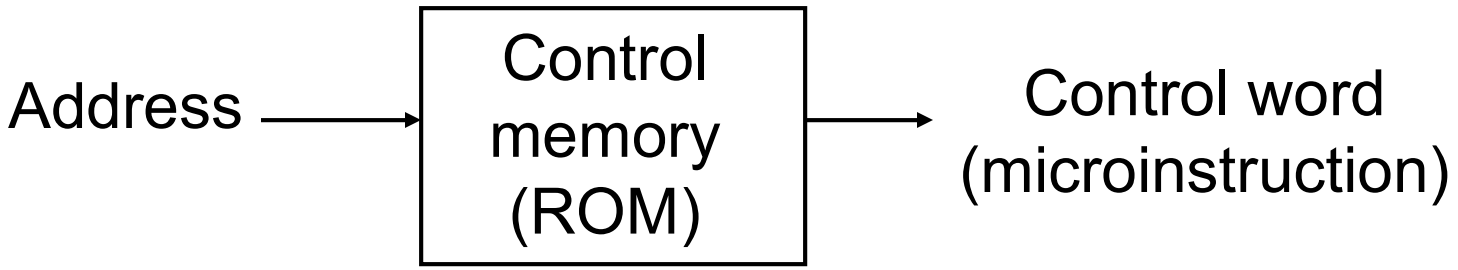
- Control memory
  - Memory contains control words
- Microinstructions
  - Control words stored in control memory
  - Specify control signals for execution of microoperations
- Microprogram
  - Sequence of microinstructions

Microprogrammed Control Unit

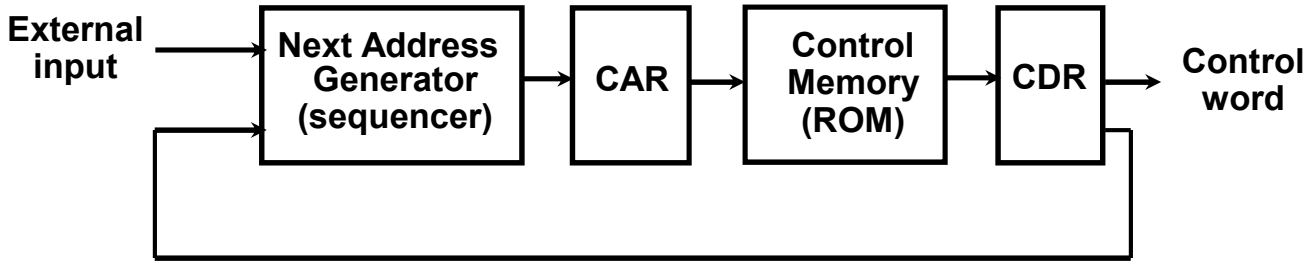
- A computer that holds a microprogrammed control unit will have two separate memories: main memory and control memory.
- The content of the main memory can be altered depending on the user requirement.
- The control memory contains fixed number of microprograms that cannot be altered by the user.

Control Memory

- Read-only memory (ROM)
- Content of word in ROM at given address specifies microinstruction
- Each computer instruction initiates series of microinstructions (microprogram) in control memory
- These microinstructions generate microoperations to
  - Fetch instruction from main memory
  - Evaluate effective address
  - Execute operation specified by instruction
  - Return control to fetch phase for next instruction



Microprogrammed Control Organization



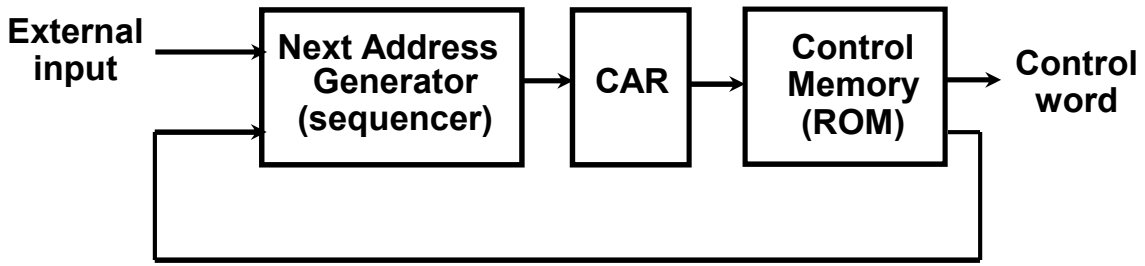
- Control memory
  - Contains microprograms (set of microinstructions)
  - Microinstruction contains
    - Bits initiate microoperations
    - Bits determine address of next microinstruction
- Control address register (CAR)
  - Specifies address of next microinstruction

Microprogrammed Control Organization

- Next address generator (microprogram sequencer)
  - Determines address sequence for control memory
- Microprogram sequencer functions
  - Load initial address into CAR to start control operations
  - Increment CAR by one
  - Transfer external address into CAR

Microprogrammed Control Organization

- Control data register (CDR)- or pipeline register
  - Holds microinstruction read from control memory
  - Allows execution of microoperations specified by control word simultaneously with generation of next microinstruction
- Control unit can operate without CDR



Microprogram Routines

- Routine
  - Group of microinstructions stored in control memory
- Each computer instruction has its own microprogram routine to generate microoperations that execute the instruction

Microprogram Routines

- Subroutine
  - Sequence of microinstructions used by other routines to accomplish particular task
- Example
  - Subroutine to generate effective address of operand for memory reference instruction
- Subroutine register (SBR)
  - Stores return address during subroutine call

Conditional Branching

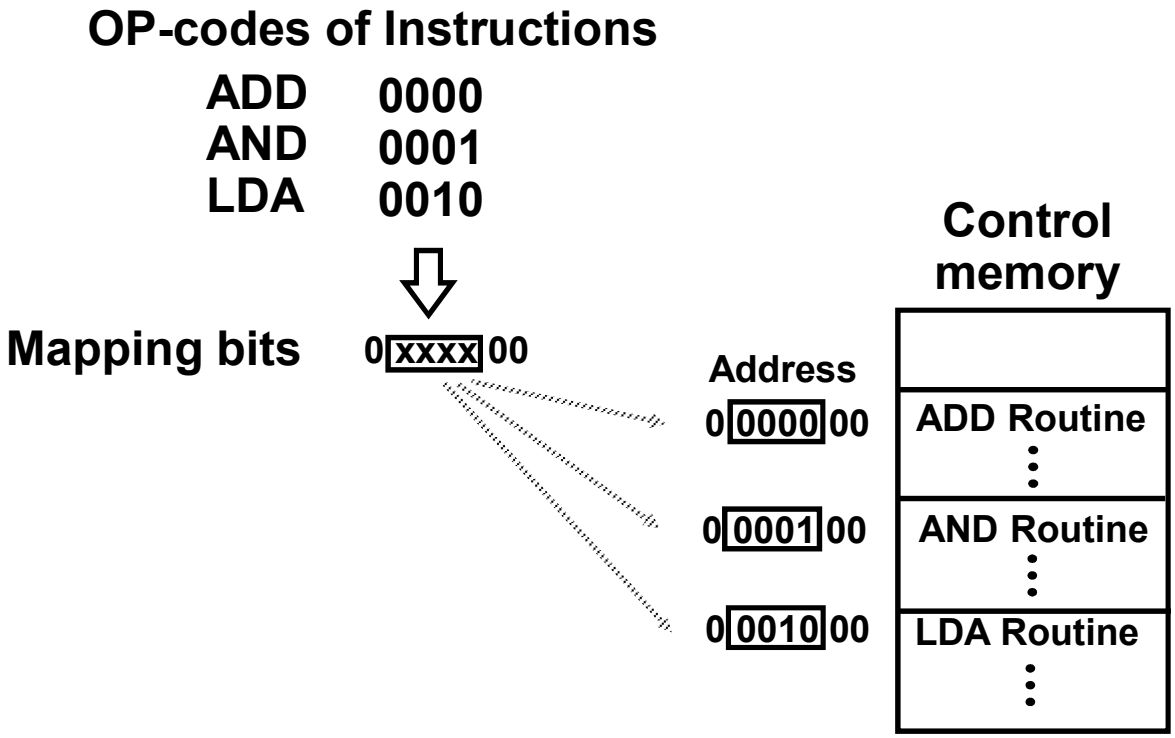
- Branching from one routine to another depends on status bit conditions
- Status bits provide parameter info such as
  - Carry-out of adder
  - Sign bit of number
  - Mode bits of instruction
- Info in status bits can be tested and actions initiated based on their conditions: 1 or 0
- Unconditional branch
  - Fix value of status bit to 1

Mapping of Instruction

- Each computer instruction has its own microprogram routine stored in a given location of the control memory
- Mapping
  - Transformation from instruction code bits to address in control memory where routine is located

Mapping of Instruction

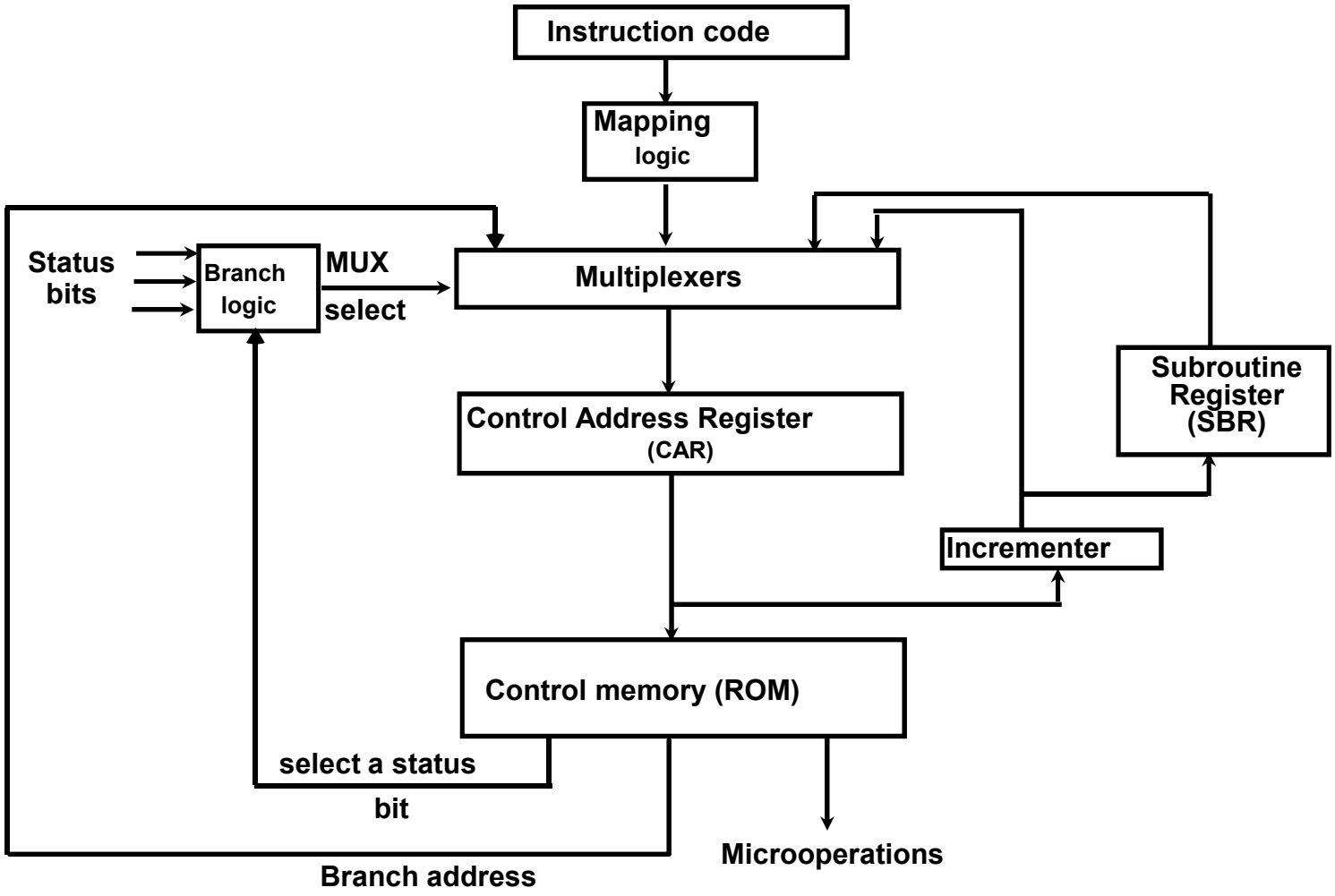
- Example
  - Mapping 4-bit operation code to 7-bit address



Address Sequencing

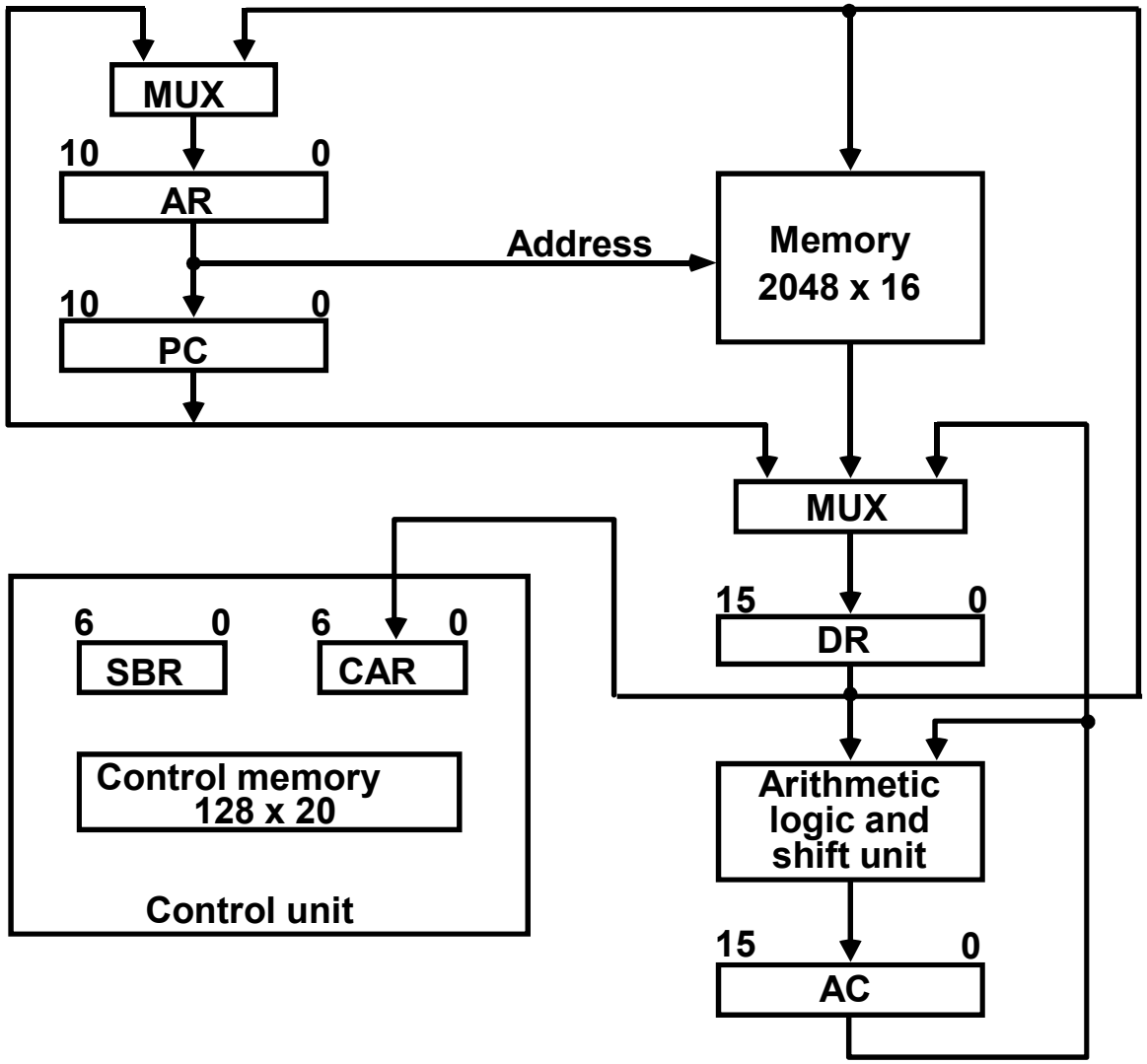
- Address sequencing capabilities required in control unit
  - Incrementing CAR
  - Unconditional or conditional branch, depending on status bit conditions
  - Mapping from bits of instruction to address for control memory
  - Facility for subroutine call and return

Address Sequencing



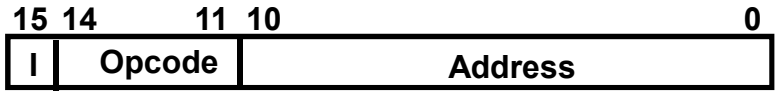
Microprogram Example

Computer Configuration



Microprogram Example

Computer instruction format

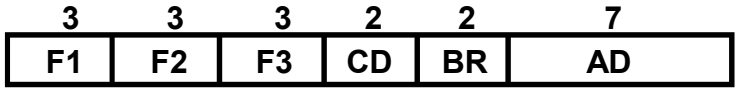


Four computer instructions

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



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

Microinstruction Fields

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

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

Microinstruction Fields

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 Microinstruction

- **Sample Format**

Label:	Micro-ops	CD	BR	AD

- Label — may be empty or may specify symbolic address terminated with colon
- Micro-ops — consists of 1, 2, or 3 symbols separated by commas
- CD — one of {U, I, S, Z}
  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}

Fetch Routine

- Fetch routine
  - Read instruction from memory
  - Decode instruction and update PC

Microinstructions for fetch routine:

```
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 fetch routine:

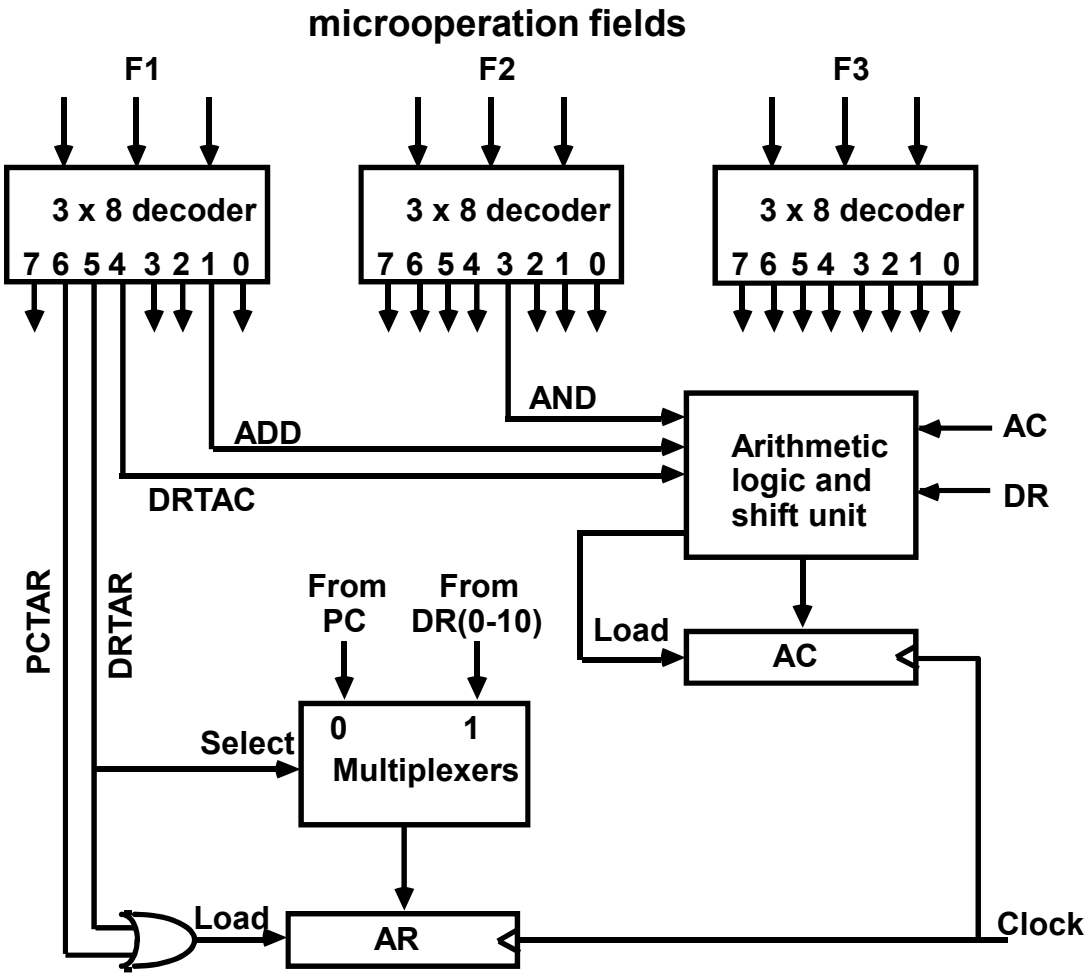
```
        ORG 64
FETCH:  PCTAR          U  JMP  NEXT
        READ, INCPC    U  JMP  NEXT
        DRTAR          U  MAP
```

Binary microporgram for fetch routine:

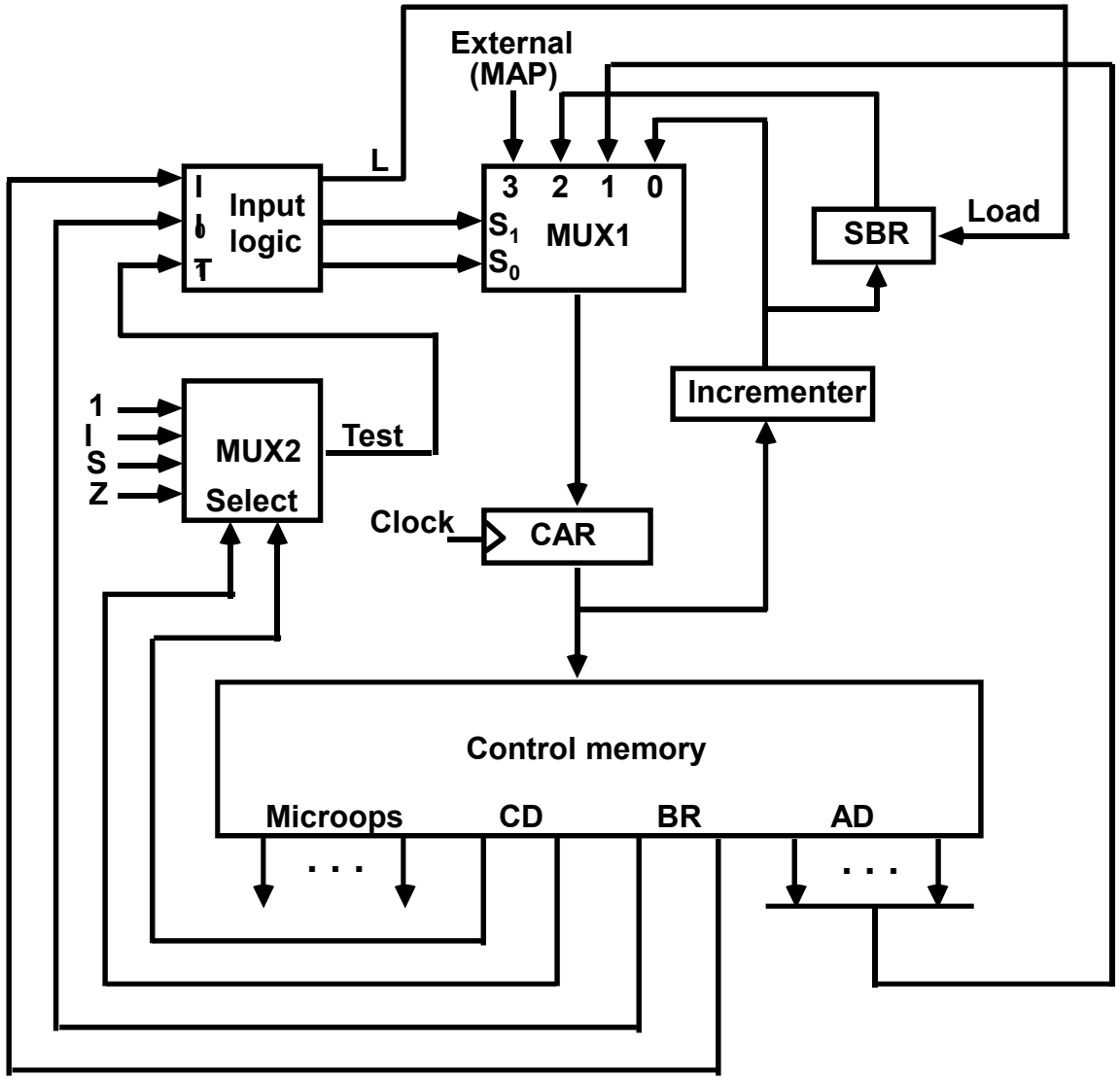
Binary address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

ORG is a pseudo-instruction used to define the origin, or first address of the microprogram routine. Here it informs the assembler to place the next microinstruction in control memory at decimal address 64.

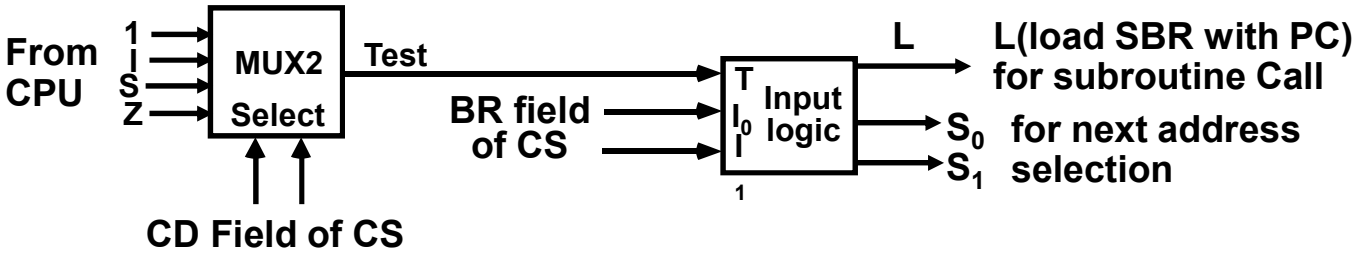
Design of Control Unit



Microprogram Sequencer



Input Logic for Microprogram Sequencer



Input Logic

I1I0T	Meaning	Source of Address	S_1S_0	L
000	In-Line	CAR+1	00	0
001	JMP	CS(AD)	01	0
010	In-Line	CAR+1	00	0
011	CALL	CS(AD) and SBR <- CAR+1	01	1
10x	RET	SBR	10	0
11x	MAP	DR(11-14)	11	0

$S1 = I1$

$S_0 = I_0I_1 + I1'T$

$L = I_1'I0T$