CHAPTER 3
CONTROL UNIT

ControlUnit

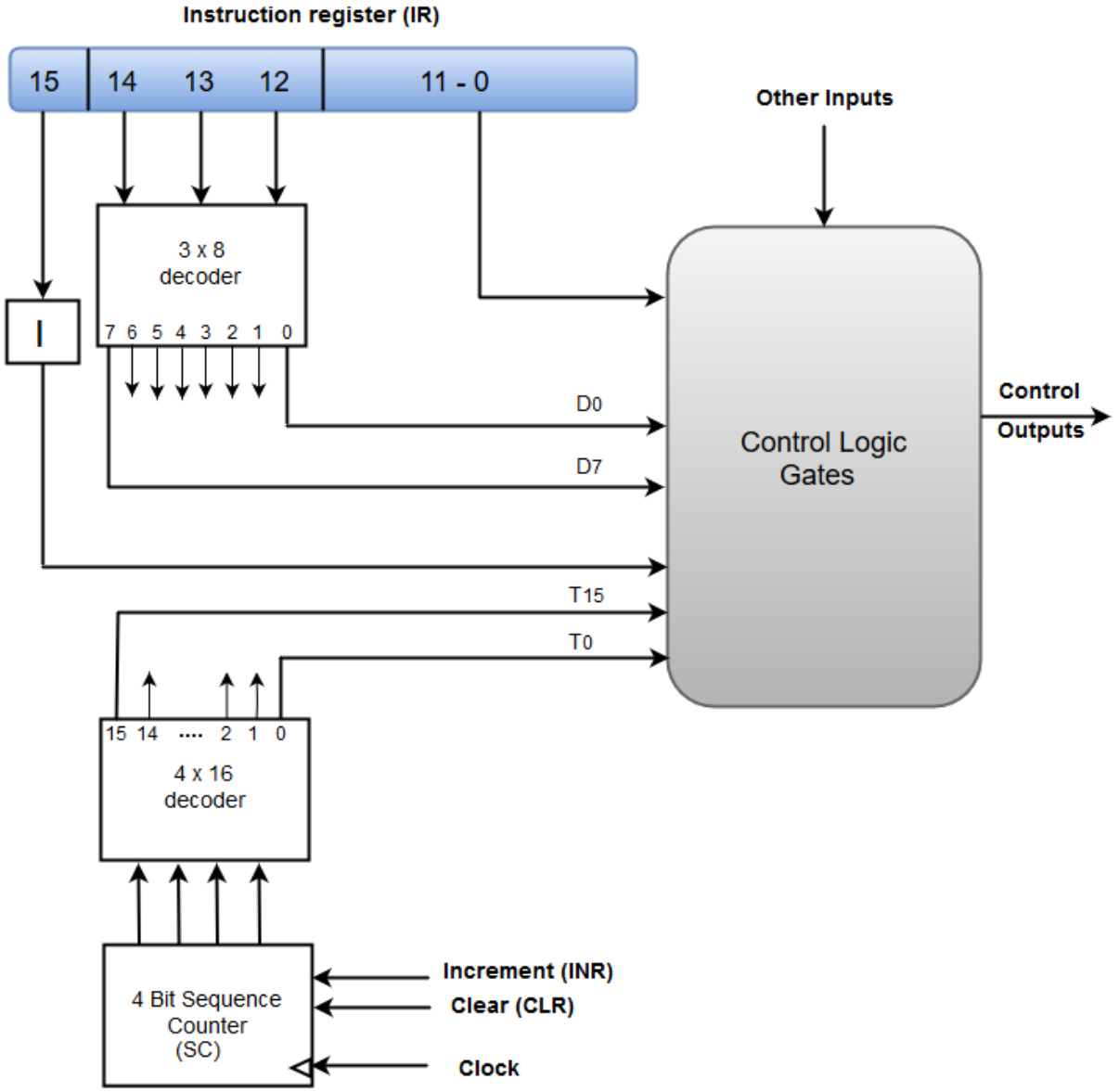
- Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them.
- The function of the control unit in a digital computer is to initiate sequences of microoperations.
- The number of different types of microoperations that are available in a given system is finite.
- The complexity of the digital system is derived from the number of sequences of microoperations that are performed.

Two techniques used for implementing control unit are **hardwired and microprogrammed control unit.**

Hardwired Control Unit	Microprogrammed Control Unit
Hardwired control unit generates the control signals needed for the processor using logic circuits	Microprogrammed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to microprogrammed control unit as the required control signals are generated with the help of hardwares	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

Control Unit of a Basic Computer (Hardwired Control)

Control Unit of a Basic Computer:



Description

- A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
- An instruction fetched from the memory unit is placed in the instruction register (IR).
- The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
- The outputs of the decoder are designated by the symbols D0 through D7.
- The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.
- The operation codes from Bits 0 through 11 are applied to the control logic gates.
- The Sequence counter (SC) can count in binary from 0 through 15.

Cont..

- **Hardwired Control:**

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

- **Microprogrammed Control:**

Microprogramming is a second alternative for designing the control unit of a digital computer which uses microoperations sequences.

A computer that employs a microprogrammed control unit will have two separate memories: **a main memory and a control memory.**

Control Memory

- **Control Memory (Control Storage: CS):** Storage in the microprogrammed control unit to store the microprogram.
- **Control word:** It is a string of control variables (0's and 1's) occupying a word in control memory.
- **Microprogram:**
- Program stored in control memory that generates all the control signals required to execute the instruction set correctly
- Consists of microinstructions

Cont..

- **Microinstruction:**

➢Contains a control word and a sequencing word

➢**Control Word** – contains all the control information required for one clock cycle

➢**Sequencing Word** - Contains information needed to decide the next microinstruction address

- **Microoperation:**

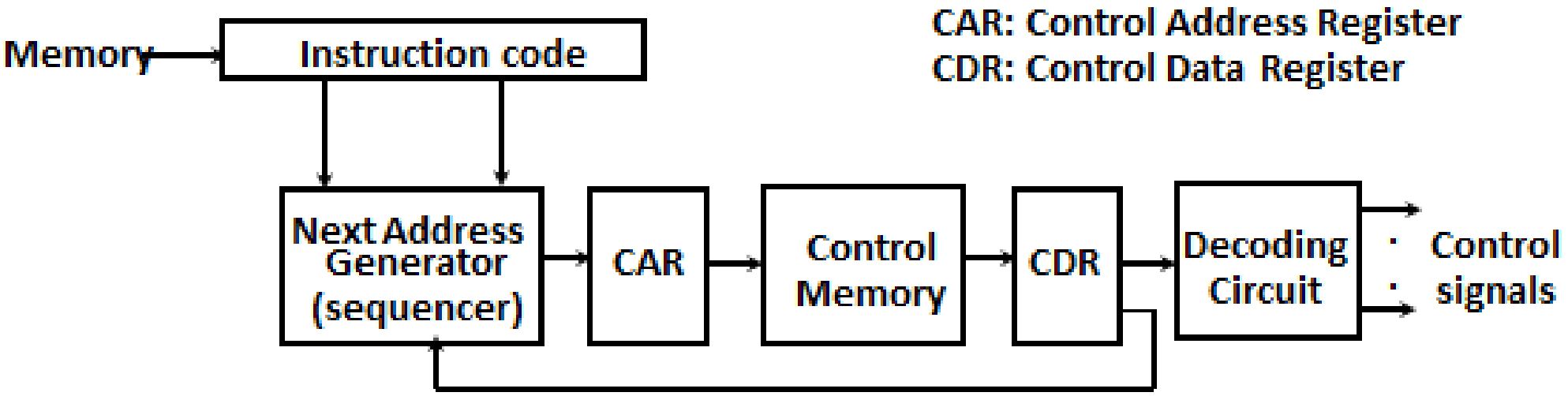
➢A microinstruction contains one or more microoperations to be completed.

Cont..

- A computer that employs a microprogrammed control unit will have two separate memories: main memory and a control memory.
- The user's program in main memory consists of machine instructions and data. The **control memory** holds a fixed microprogram that cannot be altered by the occasional user. The microprogram consists of microinstructions that specify various internal control signals for execution of register microoperations

The general configuration of a microprogrammed control unit is demonstrated in the following block diagram:

Block Diagram of Microprogrammed control unit



Description

- A more advanced development that permits a microprogram to be loaded initially from an auxiliary memory such as magnetic disk.
- Computer system whose control unit is implemented with a microprogram in WCS.
- Microprogram can be changed by a systems programmer or a user.

Sequencer: The device or program that generates address of next microinstruction to be executed is called sequencer. While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.

Cont..

- **Control Address Register:** CAR contains address of microinstruction.
- **Control Data Register:** CDR contains microinstruction read from memory. The microinstruction contains a control word that specifies one or more microoperations. The data register is sometimes called a *pipeline register.*

It allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction. This configuration requires a two-phase clock, with one clock applied to the address register and the other to the data register.

Address Sequencing

Each computer instruction has its own microprogram routine in control memory to generate the microoperations that execute the instruction. Process of finding address of next microinstruction to be executed is called ***address sequencing.*** The address sequencing capabilities required in a control memory are:

a) Incrementing of the control address register.

b) Unconditional branch or conditional branch, depending on status bit conditions.

c) A mapping process from the bits of the instruction to an address for control memory.

d) A facility for subroutine call and return.

Following is the block diagram for control memory and the associated hardware needed for selecting the next microinstruction address.

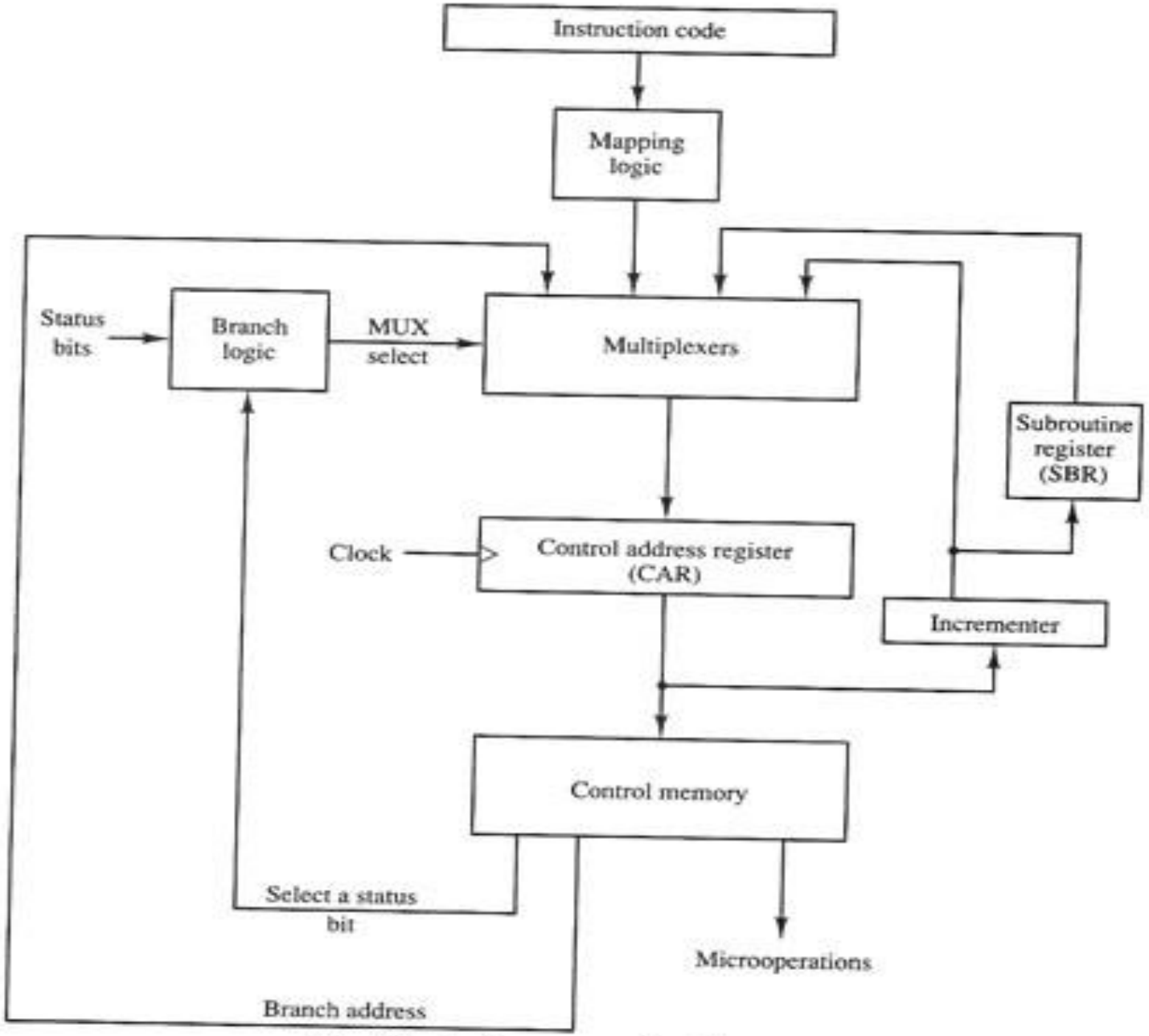


Fig: Block diagram of address sequencer.

- The diagram shows four different paths from which the control address register (CAR) receives the address.
- The incrementer increments the content of the control address register by one, to select the next microinstruction in sequence.
- Branching is achieved by specifying the branch address in one of the fields of the microinstruction.
- Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition.
- An external address is transferred into control memory via a mapping logic circuit.
- The return address for a subroutine is stored in a special register whose value is then used when the microprogram wishes to return from the subroutine.

- **Conditional Branch:**

Simplest way of implementing branch logic hardware is to test the specified condition and branch to the indicated address if condition is met otherwise address register is simply incremented. If Condition is true, hardware set the appropriate field of status register to 1. Conditions are tested for O (overflow), N (negative), Z (zero), C (carry), etc.

- **Unconditional Branch:**

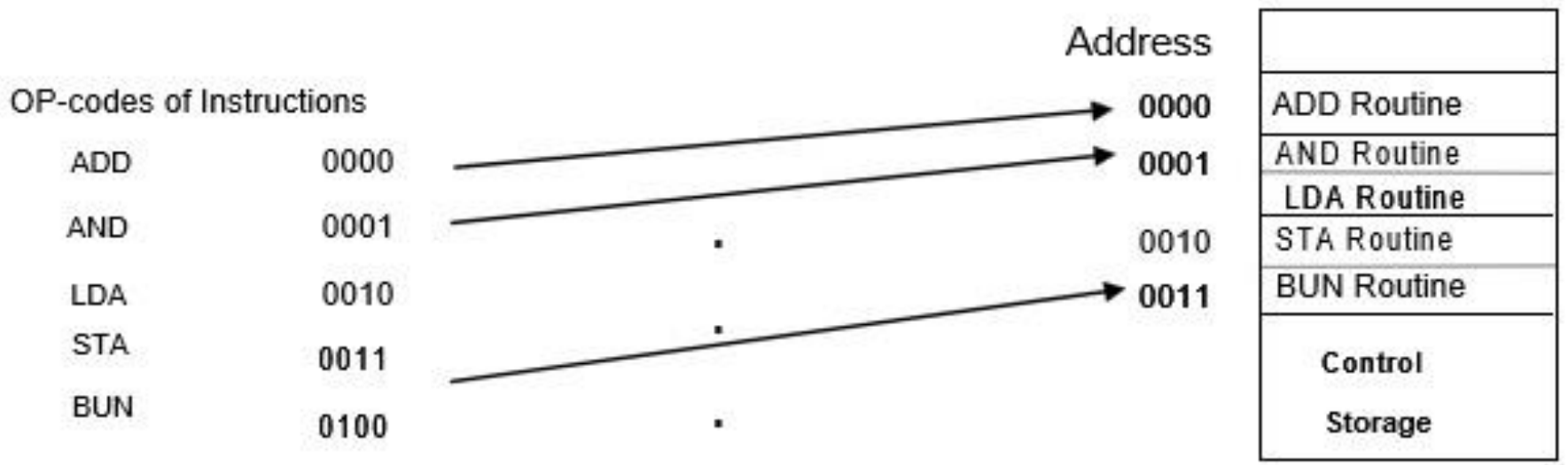
Fix the value of one status bit at the input of the multiplexer to 1. So that, branching can always be done.

- **Mapping**

Assuming operation code of 4-bits which can specify 16 (2^4) distinct instructions. Assume further and control memory has 128 words, requiring an address of 7-bits. Now we have to map 4-bit operation code into 7-bit control memory address. Thus, we have to map Op-code of an instruction to the address of the Microinstruction which is the starting microinstruction of its subroutine in memory.

- **Direct mapping:**

Directly use op-code as address of Control memory



- **Another approach of direct mapping:**

Transfer Op-code bits to use it as an address of control memory. In this mapping, one 0 is placed in the MSB and two 0s in the LSB as shown in figure:

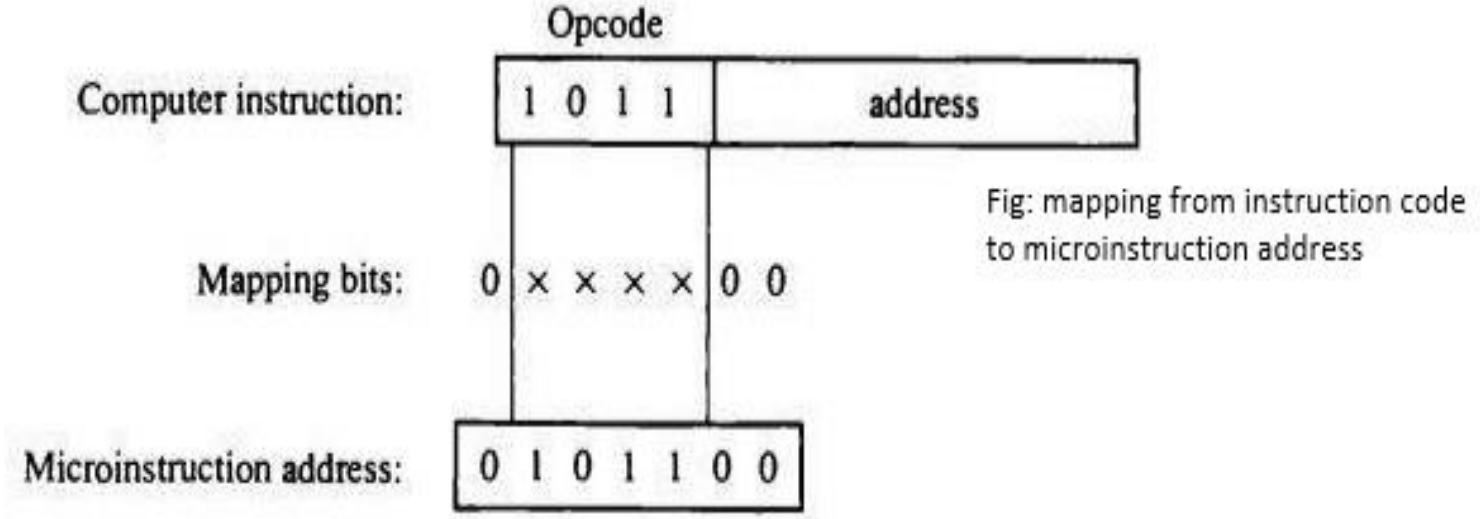
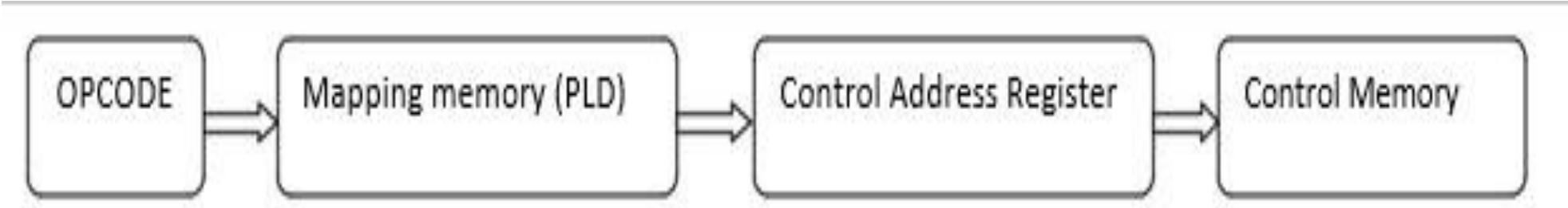


Fig: mapping from instruction code to microinstruction address

- **Extended idea: Mapping function implemented by ROM or PLD (Programmable Logic Device)**

Use op-code as address of ROM where address of control memory is stored and then use that address as an address of control memory. This provides flexibility to add instructions for control memory as the need arises.



Subroutines:

- Subroutines are programs that are used by another program to accomplish a particular task. Microinstructions can be saved by employing subroutines that use common sections of micro code.
- Example: the sequence of microoperations needed to generate the effective address is common to all memory reference instructions. Thus, this sequence could be a subroutine that is called from within many other routines to execute the effective address computation.

Subroutine register is used to save a return address during a subroutine call which is organized in LIFO (last in, first out) stack.

Computer Configuration:

- It consists of two memory units: a main memory for storing instructions and data, and a control memory for storing the microprogram
- Four registers are associated with the processor unit and two with the control unit. The processor registers are PC, AR, DR and AC.
- The control unit has control address register CAR and subroutine register SBR.
- The transfer of information among the registers in processor is done through multiplexer rather than a common bus. DR can receive information from AC, PC or memory. AR can receive information from PC or DR. PC can receive information only from AR.
- The arithmetic, logic and shift unit performs microoperations with data from AC and DR and places the result in AC. Note that memory receives its address from AR. Input data written to memory come from DR, and data read from memory can go only to DR.

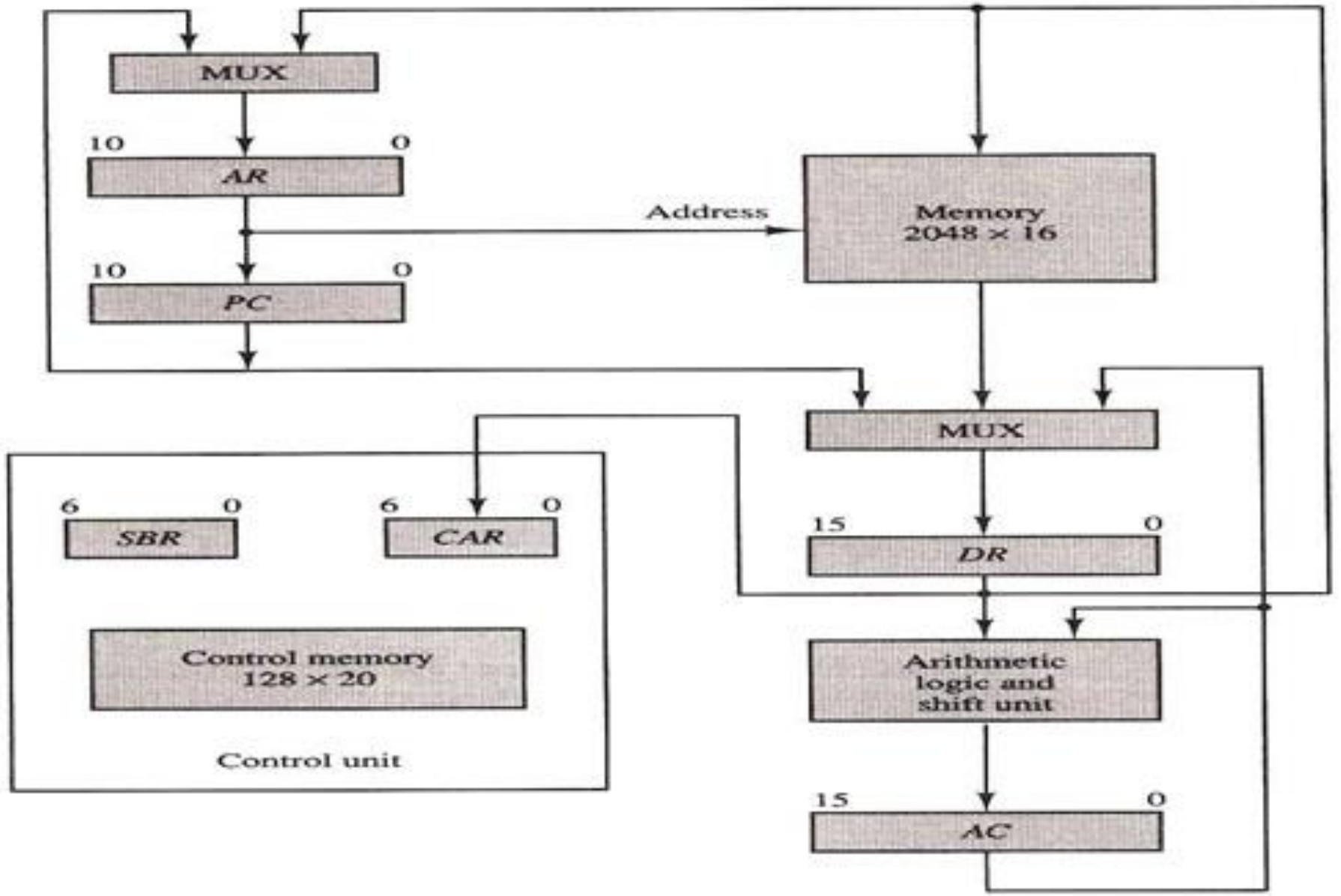
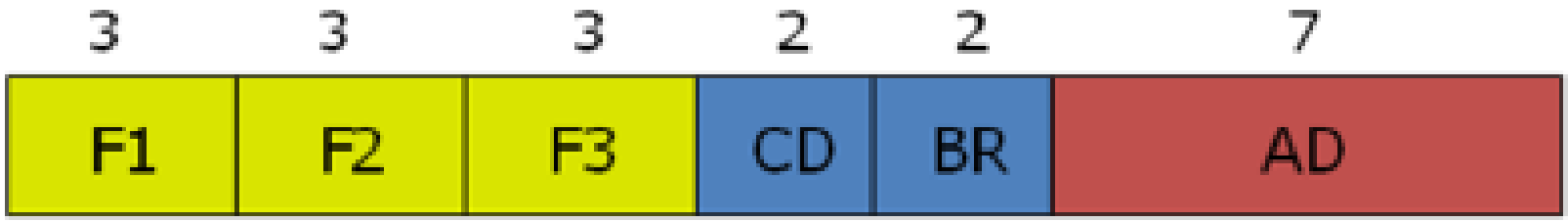


Fig: Computer hardware configuration

Microinstruction Format and Description

- **Microinstruction Format**

The 20 bits of the microinstruction are divided into four functional parts. The three fields F1, F2, and F3 specify microoperations for the computer. The CD field selects status bit conditions. The BR field specifies the type of branch to be used. The AD field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 2^7 words.



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

Fig 3-6: Microinstruction code format

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	

Cont..

CD	Condition	Symbol	Comments
00	Always = 1	U	Uncondition 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

Table 3-1 : Symbols and Binary code for Microinstruction Fields

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.
- Format of Microinstruction:

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

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

Cont…

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 (in case of MAP and RET)}

Symbolic Microprogram (example)

FETCH Routine: During FETCH Read an instruction from memory and decode the instruction and update PC

- Sequence of microoperations in the *fetch cycle*:

$$AR \leftarrow PC$$

$$DR \leftarrow \mathrm{M}[AR], \quad PC \leftarrow PC + 1$$

$$AR \leftarrow DR(0\text{–}10), \quad CAR(2\text{–}5) \leftarrow DR(11\text{–}14), \quad CAR(0,1,6) \leftarrow 0$$

Symbolic microprogram for the fetch cycle:

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

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

Binary Microprogram

END of CHAPTER 3