MICROPROGRAMMED CONTROL

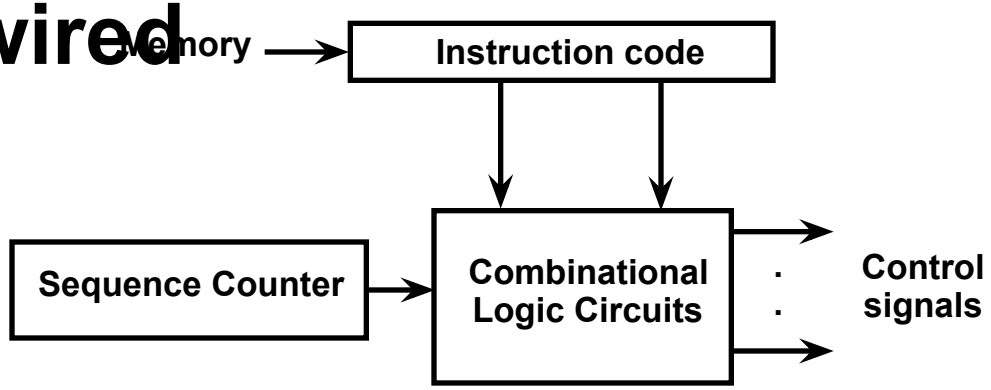
- Control Memory
- Sequencing Microinstructions
- Microprogram Example
- Design of Control Unit
- Microinstruction Format

Control Memory

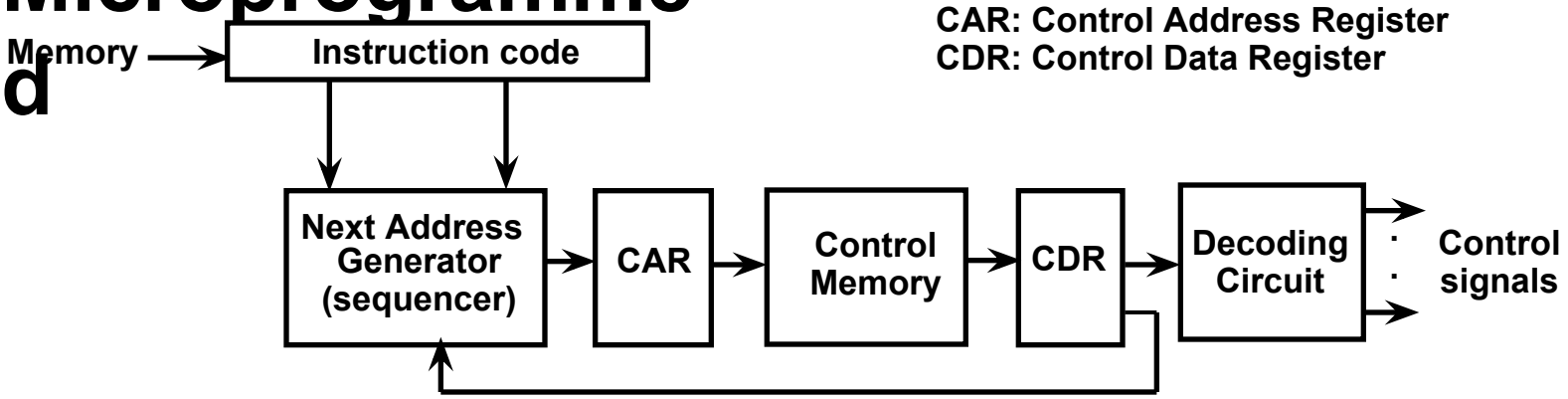
- When control signals are generated by hardware using conventional logic design techniques, the control unit is ***hardwired***.
- ***Microprogramming*** is an alternative for designing the control unit.
- The control function is a binary variable and if 1, then that microoperation is executed.
- In a bus-organized system, the control signals that specify microoperations are group of bits that select the paths in multiplexers, decoders, and arithmetic logic units.
- The control variables at any given time can be represented by a string of 1's and 0's and is called a control word.
- A control unit whose binary control variables are stored in control memory is a micrprogrammed control unit.

Control Unit Implementation

- **Hardwired**



- **Microprogrammed**



HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micrprogrammed 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
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

- **Hardwired Control Unit: When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired.**
- **Micro programmed control unit: A control unit whose binary control variables are stored in memory is called a micro programmed control unit.**

- **Dynamic microprogramming:**

A more advanced development known as dynamic microprogramming permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk. Control units that use dynamic microprogramming employ a writable control memory. This type of memory can be used for writing.

- **Control Memory: Control Memory is the storage in the microprogrammed control unit to store the microprogram.**
- **Writeable Control Memory: Control Storage whose contents can be modified, allow the change in microprogram and Instruction set can be changed or modified is referred as Writeable Control Memory.**

Microoperation, Microinstruction, Micro program, Microcode.

- Microoperations: In computer central processing units, micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed macro-instructions in this context).
- Micro instruction: A symbolic microprogram can be translated into its binary equivalent by means of an assembler. Each line of the assembly language microprogram defines a symbolic microinstruction. Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD.

Contd..

- **Micro program: A sequence of microinstructions constitutes a microprogram. Since alterations of the microprogram are not needed once the control unit is in operation, the control memory can be a read-only memory (ROM). ROM words are made permanent during the hardware production of the unit. The use of a micro program involves placing all control variables in words of ROM for use by the control unit through successive read operations. The content of the word in ROM at a given address specifies a microinstruction.**

Contd..

- **Microcode: Microinstructions can be saved by employing subroutines that use common sections of microcode. For example, the sequence of micro operations needed to generate the effective address of the operand for an instruction is common to all memory reference instructions. This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.**

Microprogrammed Control Unit

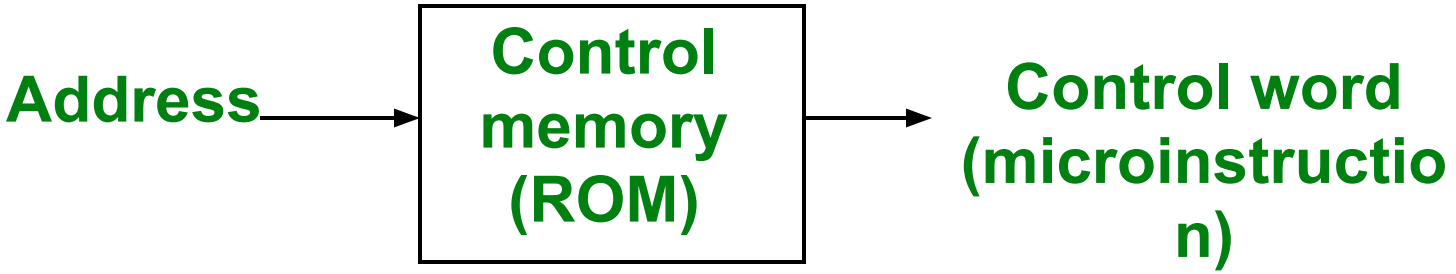
- Control signals
  - Group of bits used to select paths in multiplexers, decoders, arithmetic logic units
- Control variables
  - Binary variables specify microoperations
    - » Certain microoperations initiated while others idle
- Control word
  - String of 1's and 0's represent control variables

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**

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



Organization of micro programmed control unit

- The general configuration of a micro-programmed control unit is demonstrated in the block diagram of Figure The control memory is assumed to be a ROM, within which all control information is permanently stored.

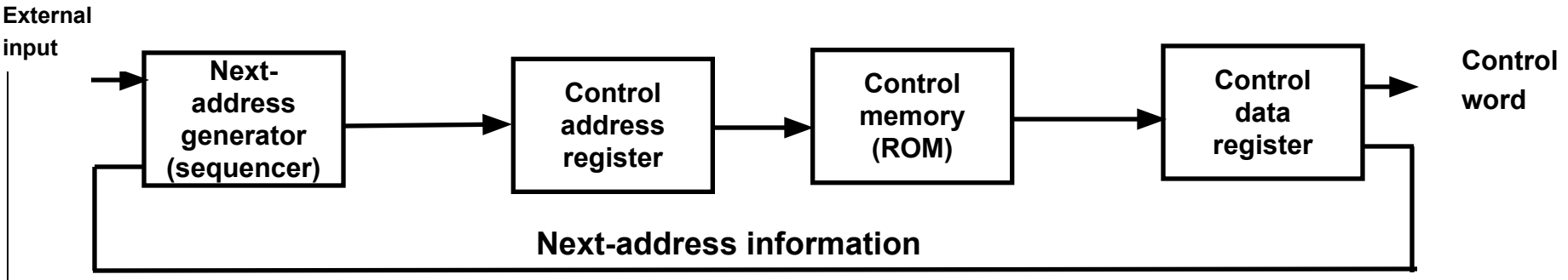
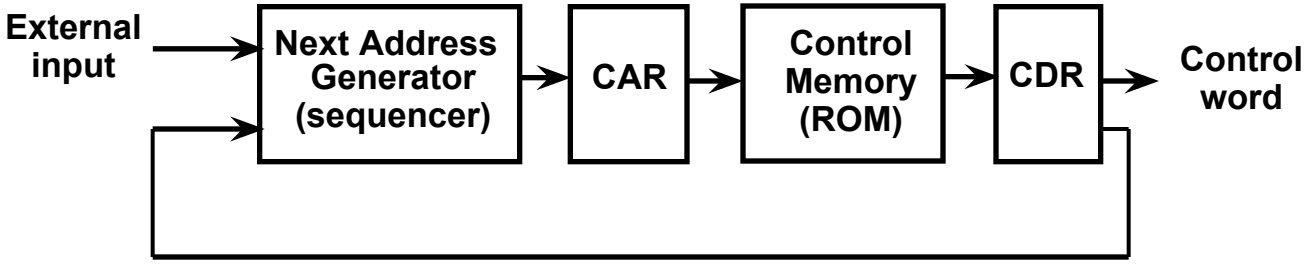


Figure Microprogrammed control organization

- **The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory. The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address. The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory**

Microprogrammed Control Organization



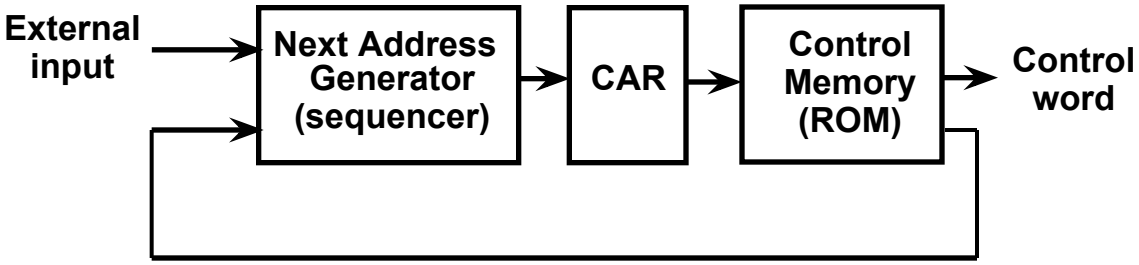
- Control memory
  - Contains microprograms (set of microinstructions)
  - Microinstruction contains
    - » Bits initiate microoperations
    - » Bits determine address of next microinstruction
- Control address register (CAR)
  - Specifies address of next microinstruction

Microprogrammed Control Organization

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

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

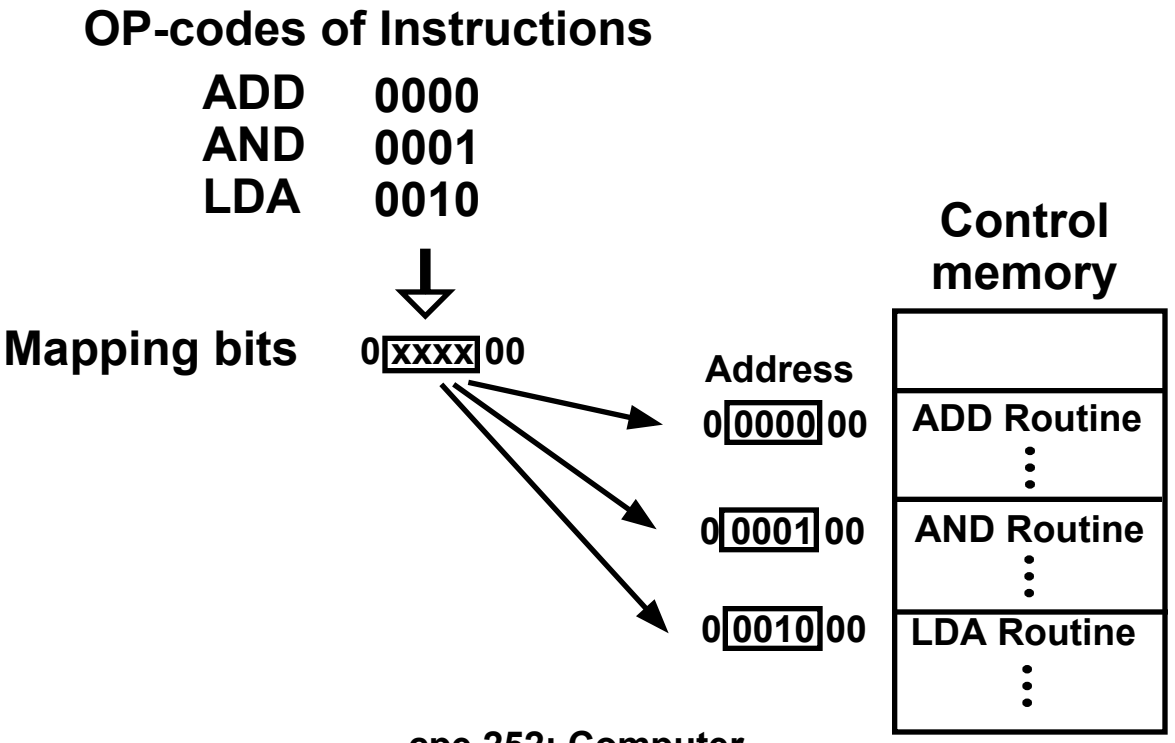
- Branching from one routine to another depends on status bit conditions
- Status bits provide parameter info such as
  - Carry-out of adder
  - Sign bit of number
  - Mode bits of instruction
- Info in status bits can be tested and actions initiated based on their conditions: 1 or 0
- Unconditional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



Mapping of Instruction

- A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located.
- The status bits for this type of branch are the bits in the operation code part of the instruction. For example, a computer with a simple instruction format as shown in figure 4.3 has an operation code of four bits which can specify up to 16 distinct instructions.
- Assume further that the control memory has 128 words, requiring an address of seven bits.

Contd..

- One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory is shown in figure 4.3.
- This mapping consists of placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits of the control address register.
- This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.
- If the routine needs more than four microinstructions, it can use addresses 1000000 through 1111111. If it uses fewer than four microinstructions, the unused memory locations would be available for other routines.

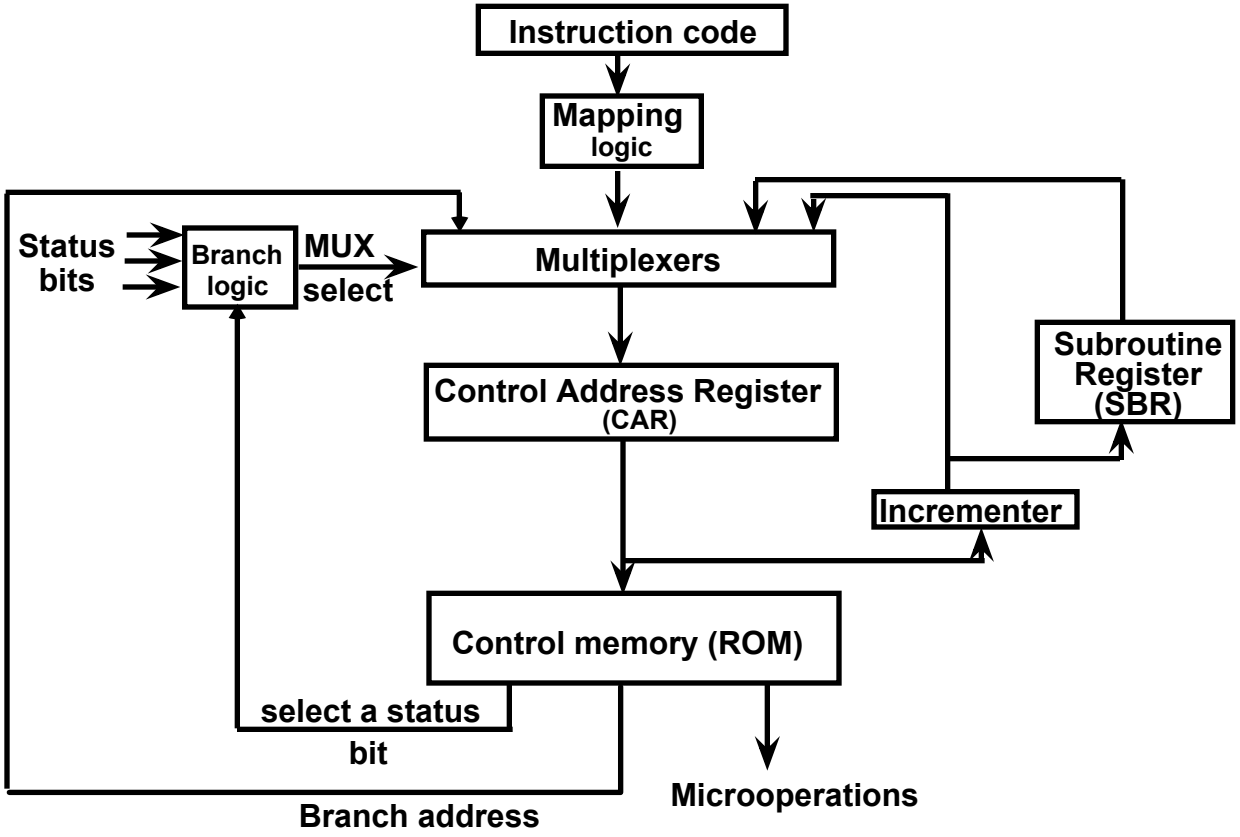
Contd..

- One can extend this concept to a more general mapping rule by using a ROM to specify the mapping function. The contents of the mapping ROM give the bits for the control address register

Address Sequencing

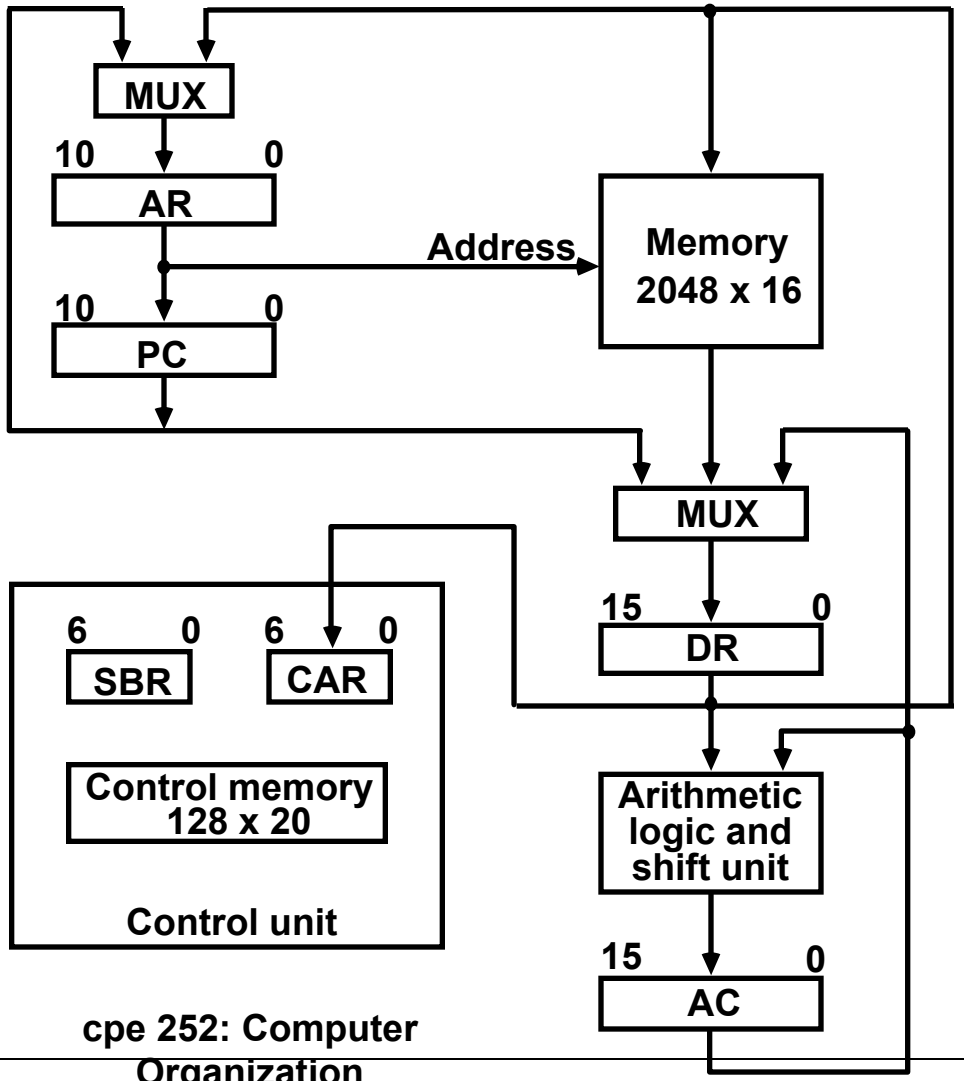
- **Address sequencing capabilities required in control unit**
  - **Incrementing CAR**
  - **Unconditional or conditional branch, depending on status bit conditions**
  - **Mapping from bits of instruction to address for control memory**
  - **Facility for subroutine call and return**

Address Sequencing



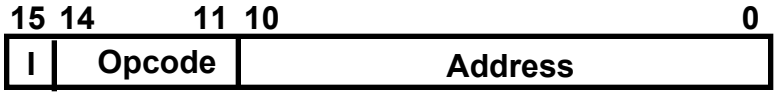
Microprogram Example

Computer Configuration



Microprogram Example

Computer instruction format

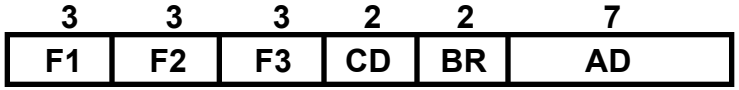


Four computer instructions

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

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 or from an auxiliary memory (as a magnetic tape)
- Microprogram can be changed by a systems programmer or a user

TERMINOLOGY

- Each control word in control memory contains within it a microinstruction
- The microinstruction specifies one or more microoperations for the system
- A sequence of microoperations constitutes a microprogram.

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

Note: It should be mentioned here that most computers on reduced instruction set computer (RISC) architecture concept use hardwired control rather than a control memory with a microprogram.

MICROINSTRUCTION SEQUENCING

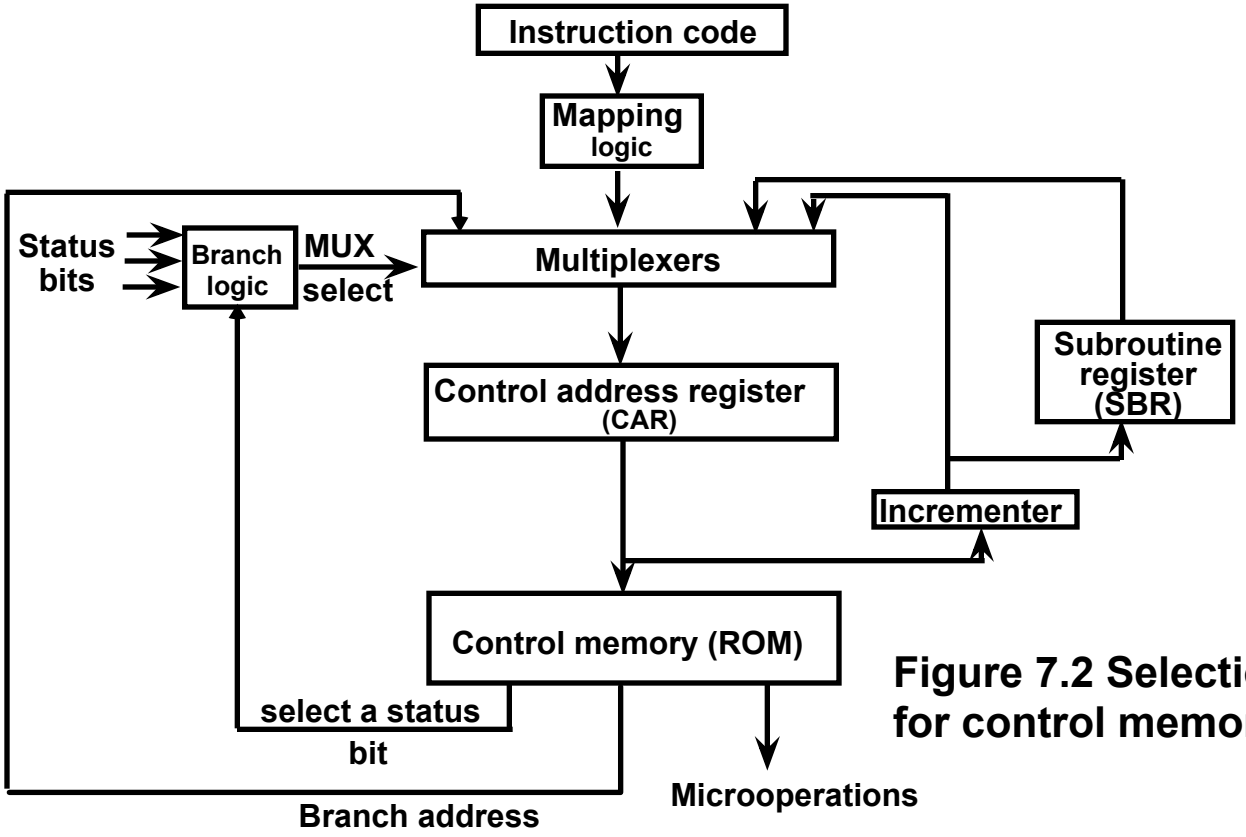
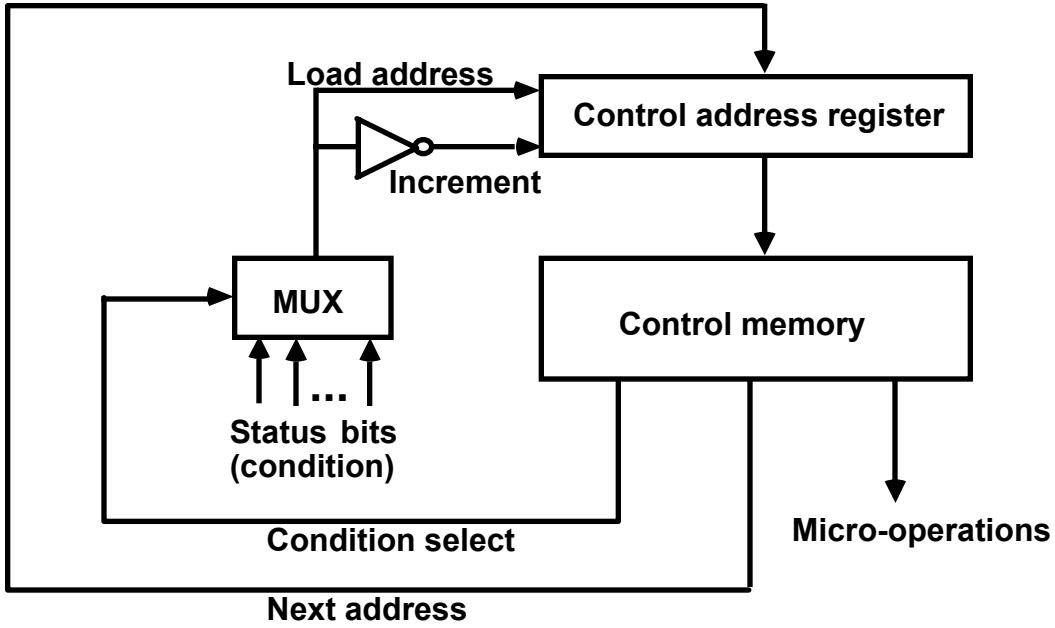


Figure 7.2 Selection of address for control memory

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

CONDITIONAL BRANCH



Conditional Branch

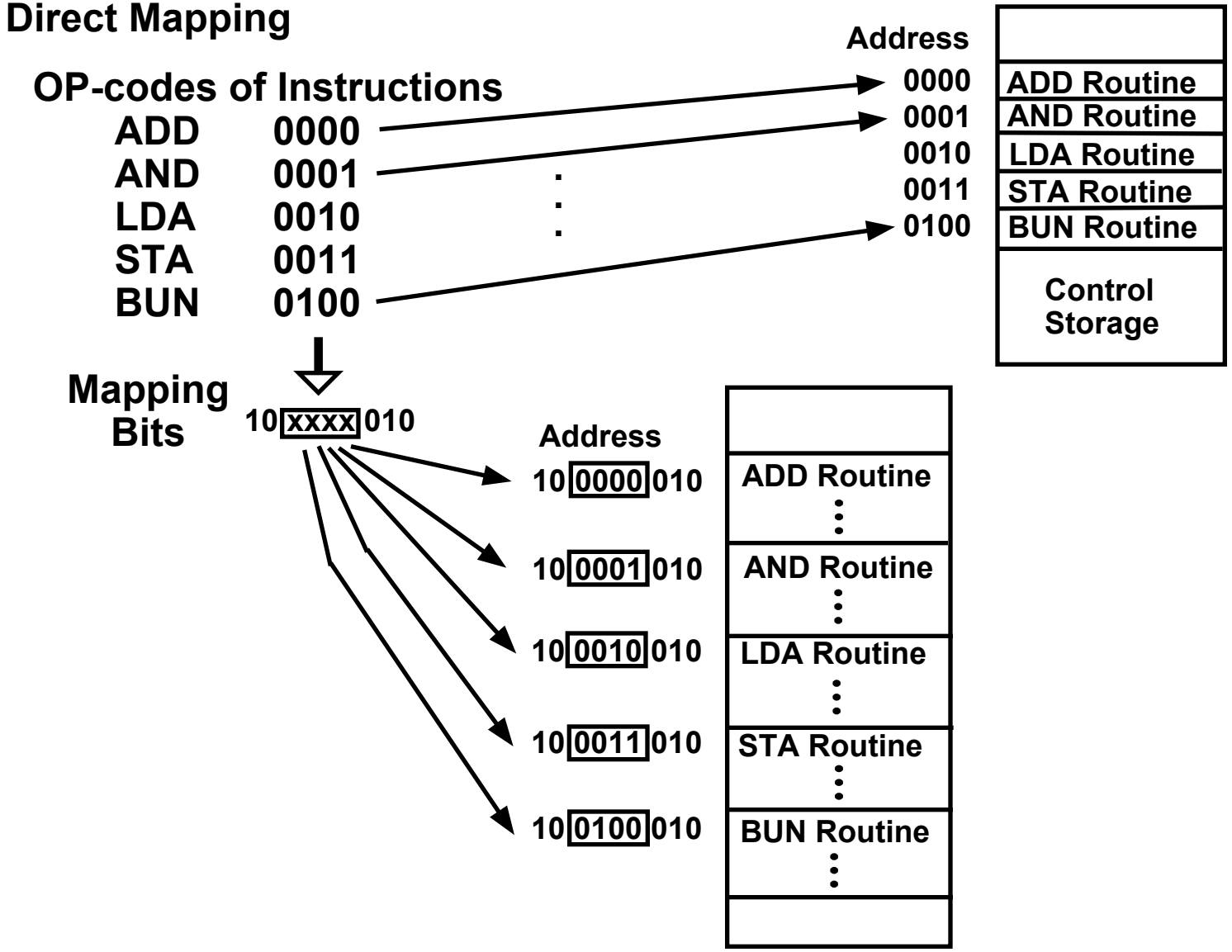
If *Condition* is true, then *Branch* (address from the next address field of the current microinstruction) else *Fall Through*

Conditions to Test: O(overflow), N(negative), Z(zero), C(carry), etc.

Unconditional Branch

Fixing the value of one status bit at the input of the multiplexer to 1

MAPPING OF INSTRUCTIONS



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

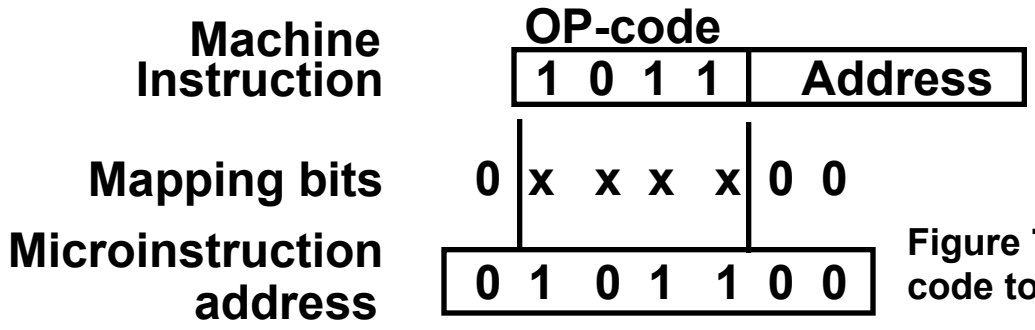
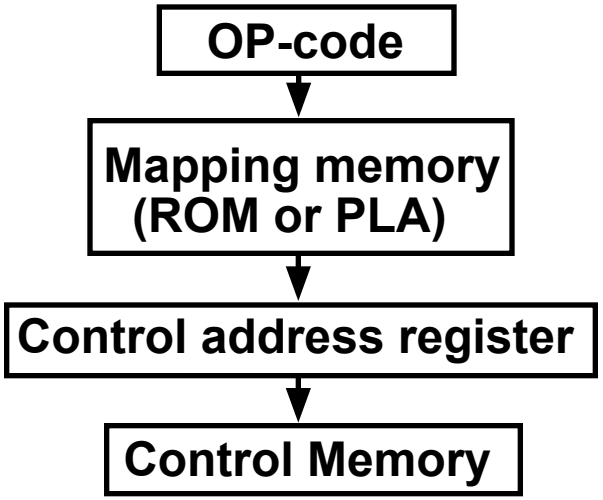


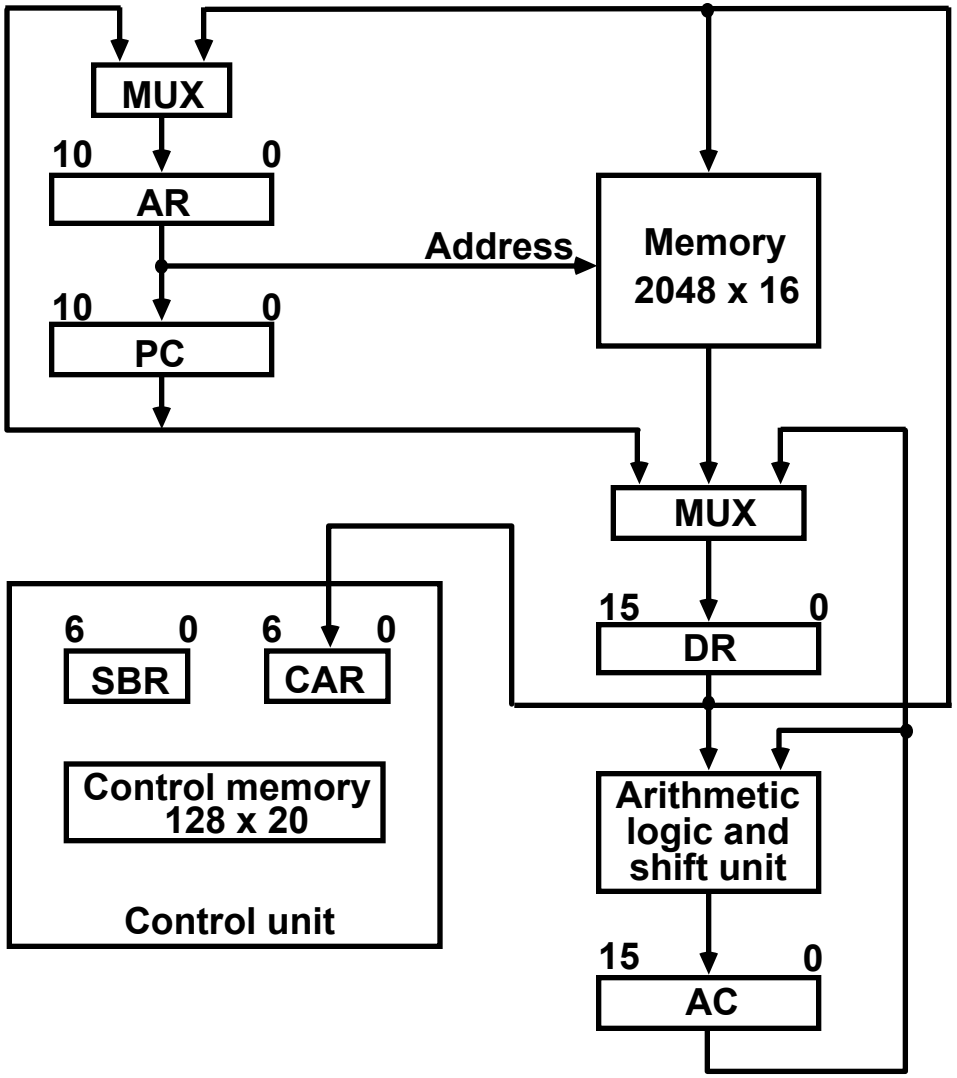
Figure 7.3 Mapping from instruction code to microinstruction address

Mapping function implemented by ROM or PLA

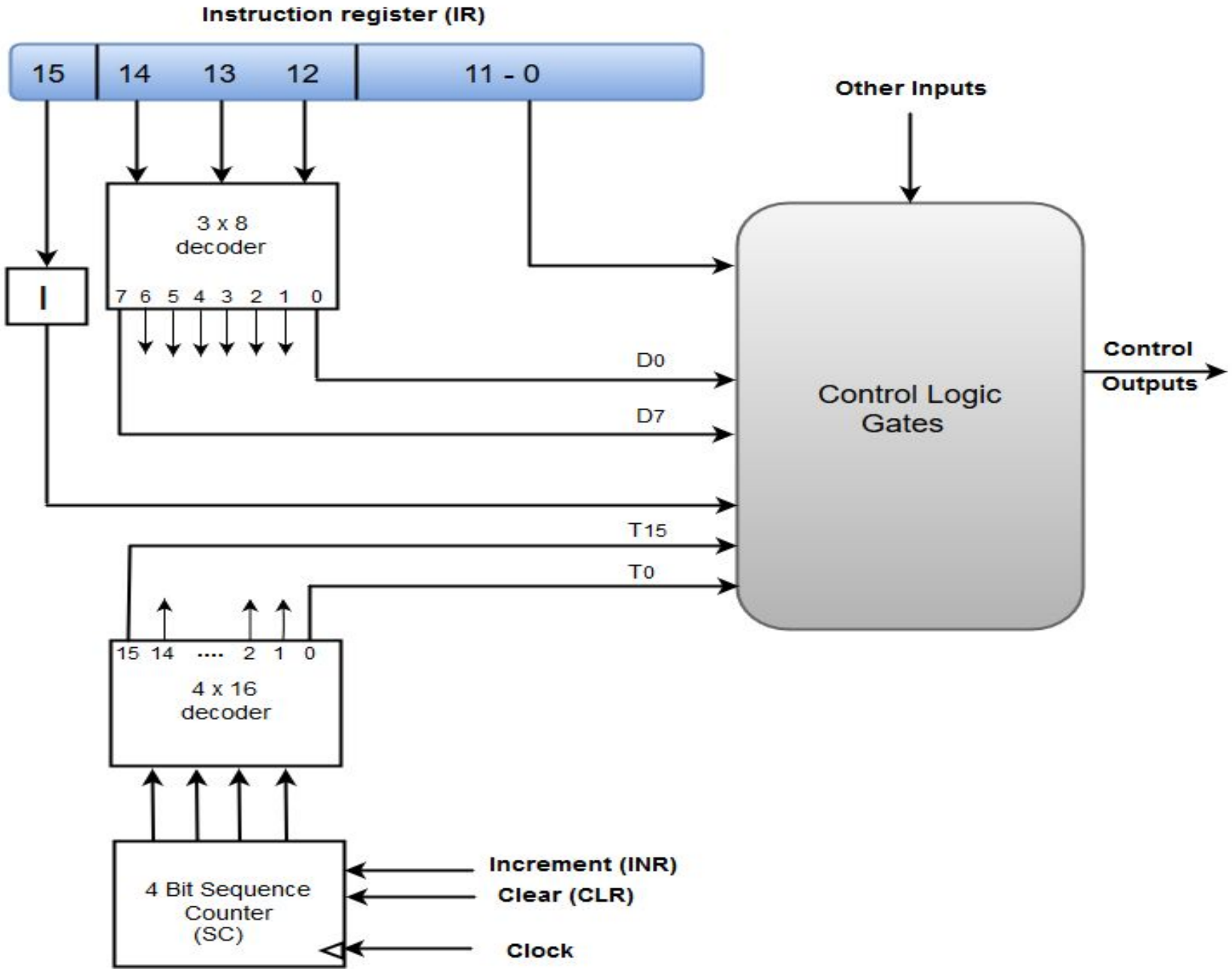


MICROPROGRAM EXAMPLE

Figure 7.4 Computer Hardware Configuration



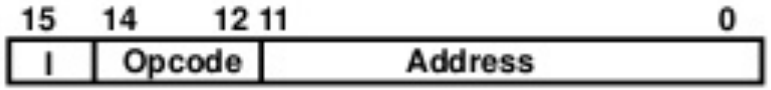
Control Unit of a Basic Computer:



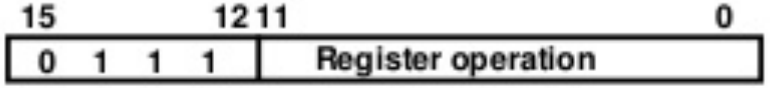
BASIC COMPUTER INSTRUCTIONS

- **Basic Computer Instruction Format**

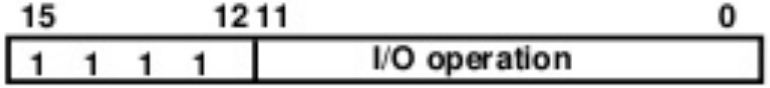
Memory-Reference Instructions (OP-code = 000 ~ 110)



Register-Reference Instructions (OP-code = 111, I = 0)



Input-Output Instructions (OP-code =111, I = 1)



DESIGN OF CONTROL UNIT
- DECODING ALU CONTROL INFORMATION -

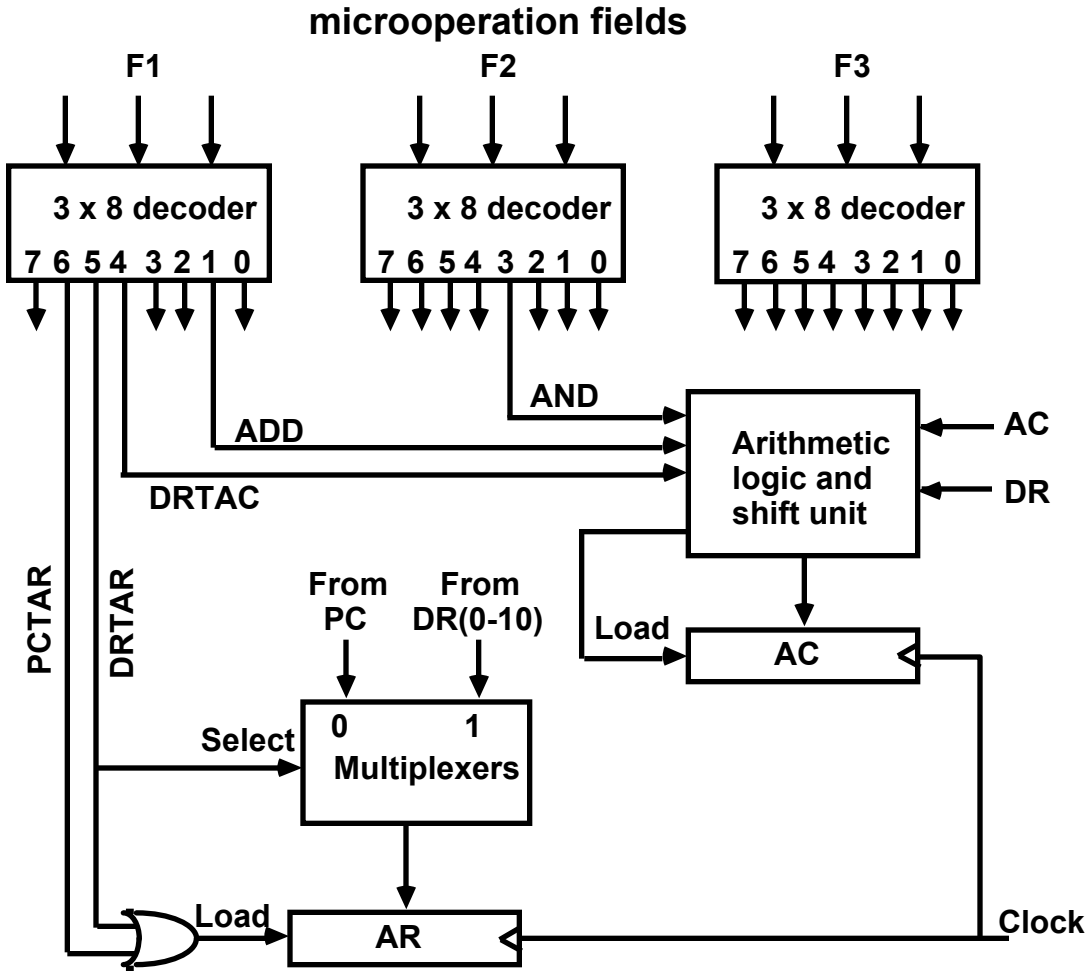
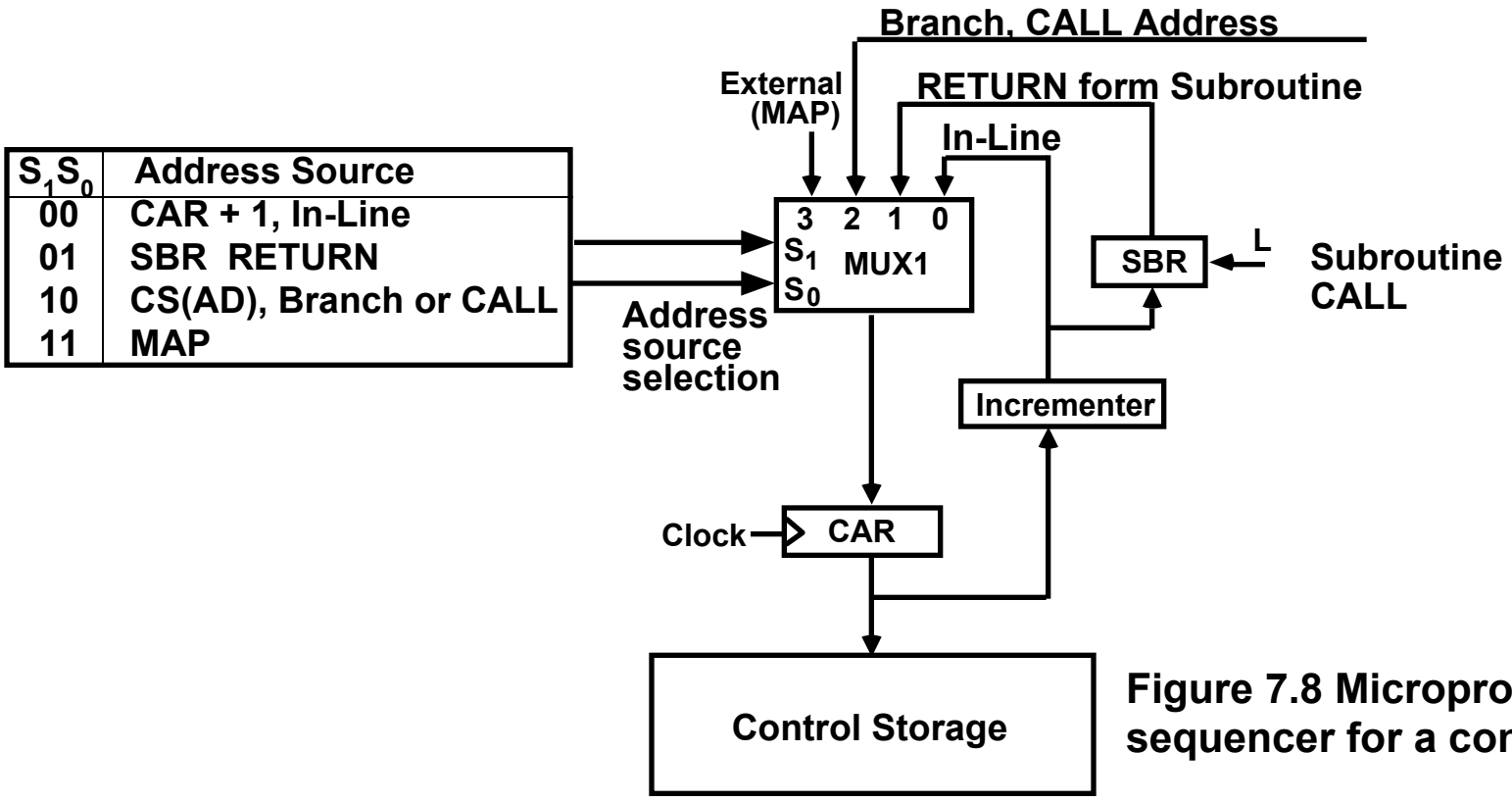


Figure 7.7 Decoding of microoperations fields

MICROPROGRAM SEQUENCER
- NEXT MICROINSTRUCTION ADDRESS LOGIC -



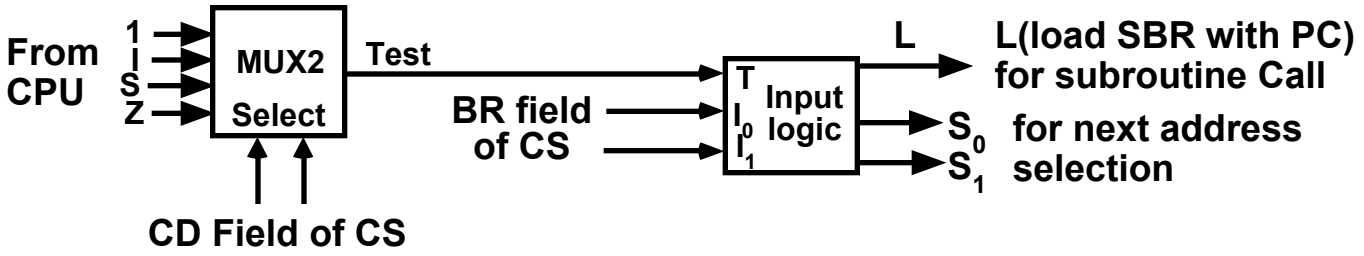
S_1S_0	Address Source
00	CAR + 1, In-Line
01	SBR RETURN
10	CS(AD), Branch or CALL
11	MAP

Figure 7.8 Microprogram sequencer for a control memory

MUX-1 selects an address from one of four sources and routes it into a CAR

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

MICROPROGRAM SEQUENCER
- CONDITION AND BRANCH CONTROL -

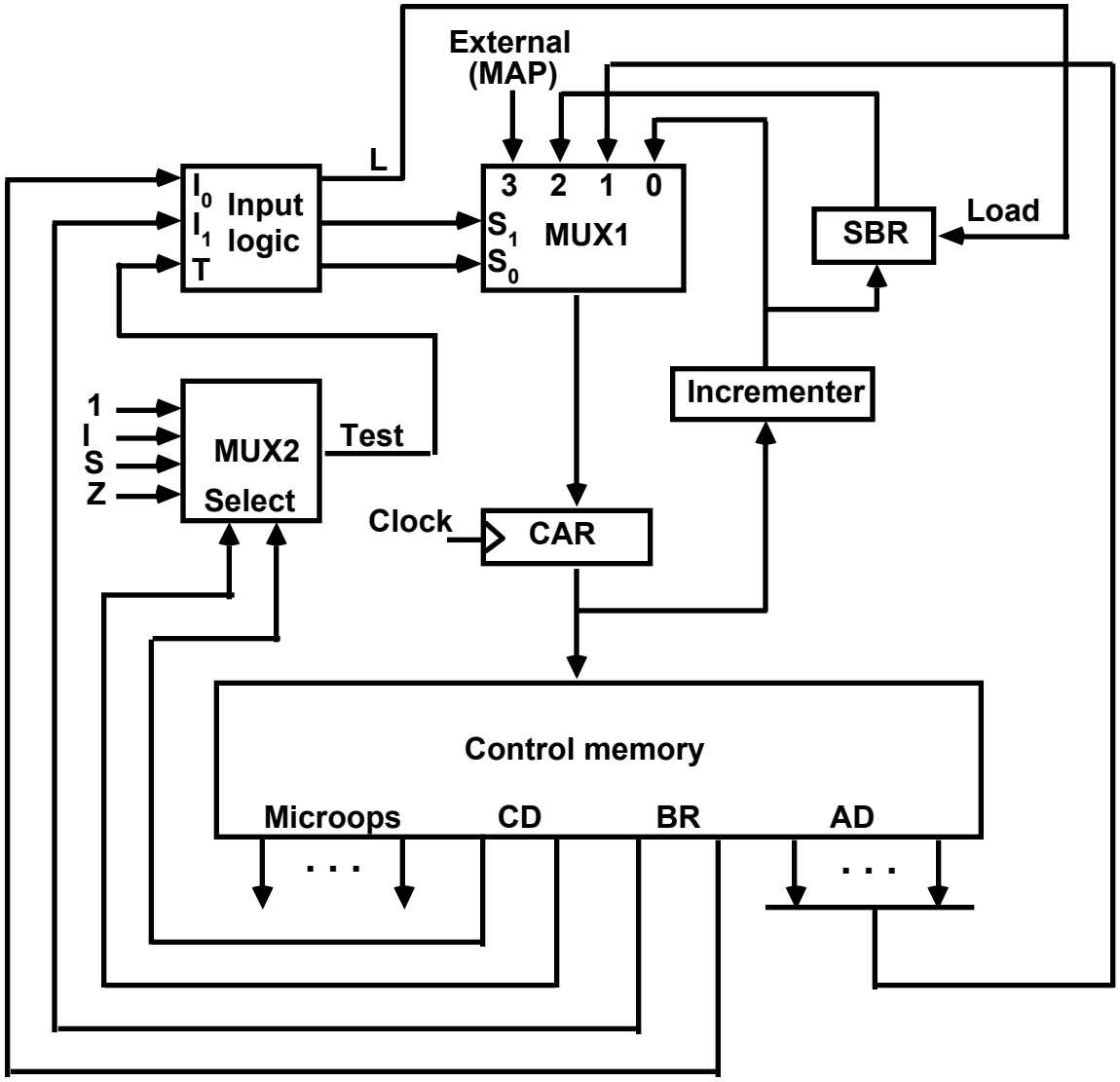


Input Logic

I_0I_1T	Meaning	Source of Address	S_1S_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

$$S_0 = I_0$$
$$S_1 = I_0I_1 + I_0'T$$
$$L = I_0'I_1T$$

MICROPROGRAM SEQUENCER



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

HORIZONTAL AND VERTICAL MICROINSTRUCTION FORMAT

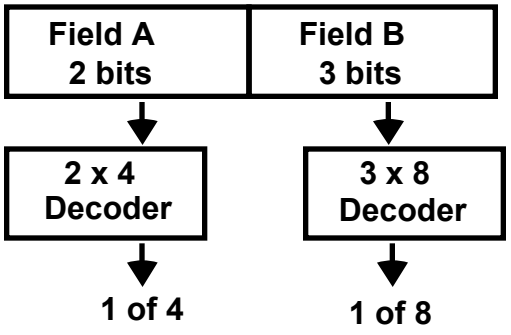
Horizontal Microinstructions

- 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

One-level decoding



Two-level decoding