SUBJECT NAME: Computer Organization & Architecture

SUBJECT CODE: 203105253

UNIT 3

Prepared By

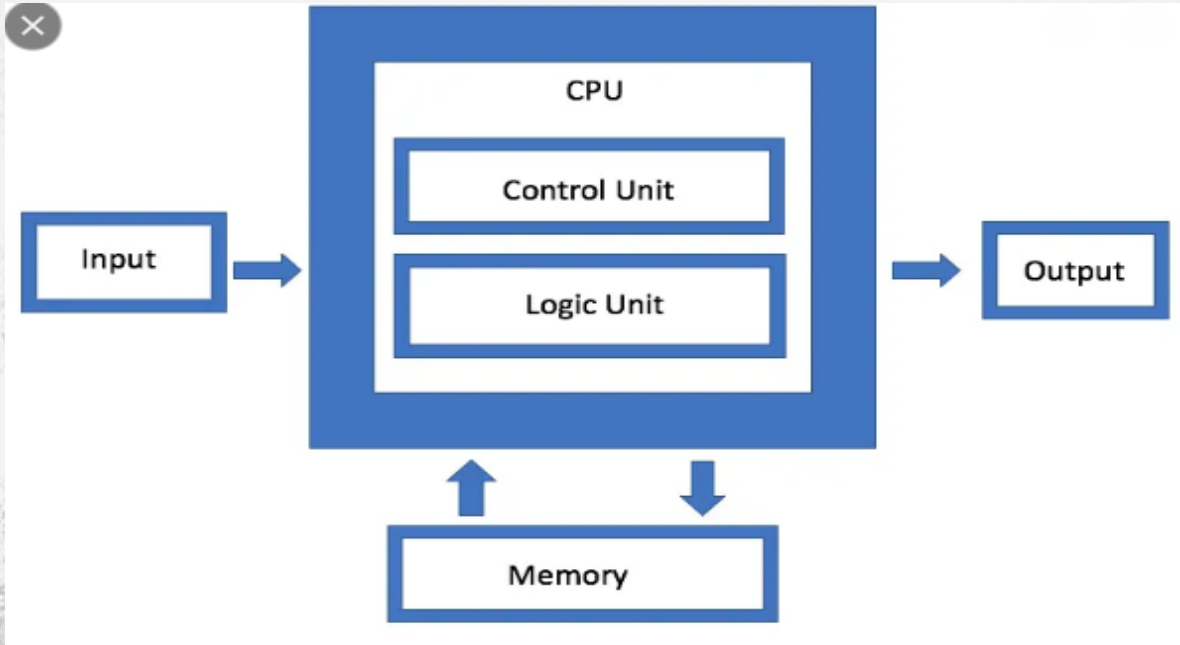
Trilok Suthar

INTRODUCTION TO X86 ARCHITECTURE:
CPU CONTROL UNIT DESIGN:
hardwired and micro-programmed design approaches, Case study -design of a simple hypothetical CPU.

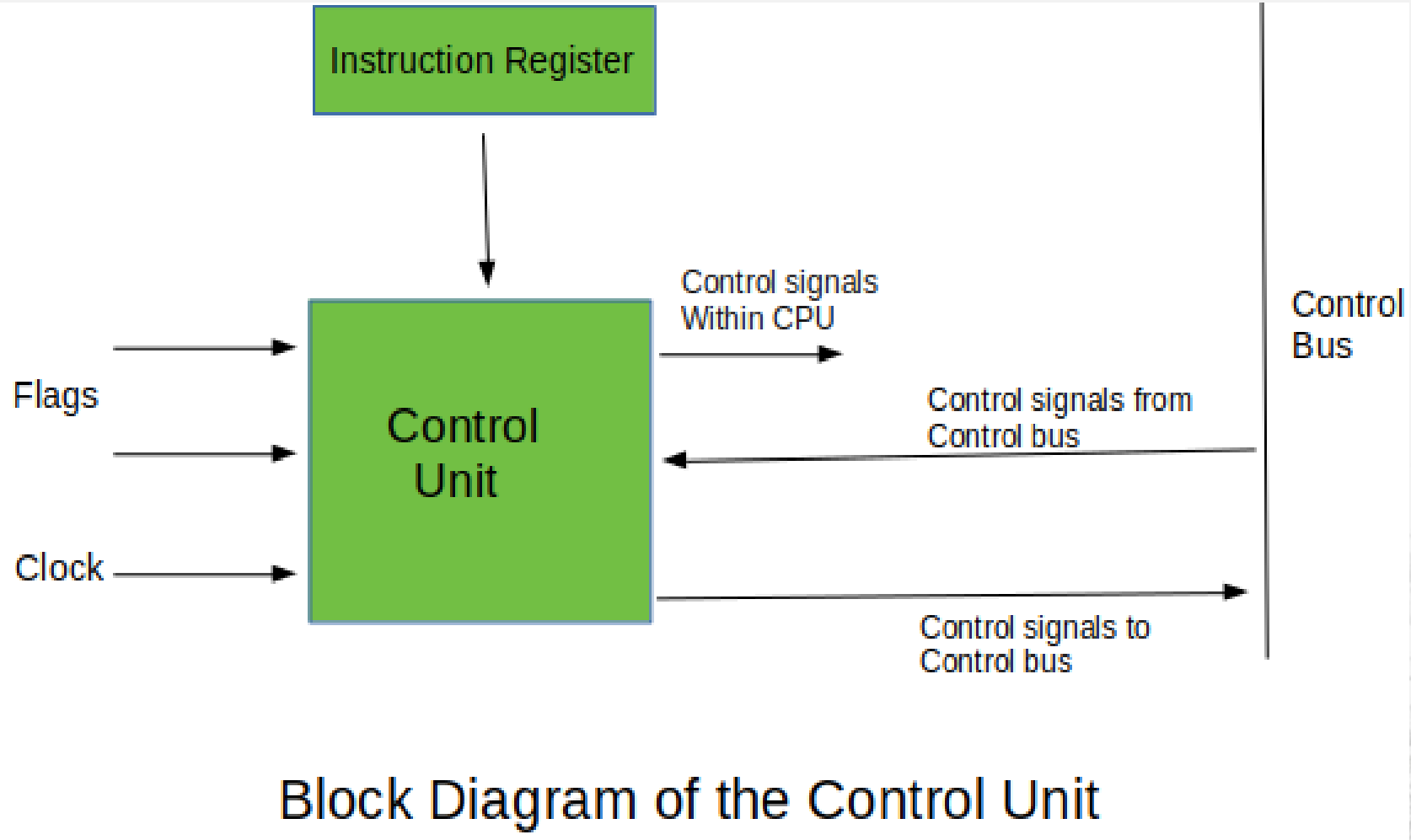
MEMORY SYSTEM DESIGN:
Semiconductor memory technologies, memory organization.

Introduction of Control Unit and its Design

- **Control Unit** is the part of the computer's central processing unit (CPU), which directs the operation of the processor. It was included as part of the Von Neumann Architecture by John von Neumann.
- It is the responsibility of the Control Unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor.



- It fetches internal instructions of the programs from the main memory to the processor instruction register, and based on this register contents, the control unit generates a control signal that supervises the execution of these instructions.
- A control unit works by receiving input information to which it converts into control signals, which are then sent to the central processor.
- The computer’s processor then tells the attached hardware what operations to perform.



Block Diagram of the Control Unit

Functions of the Control Unit –

- It coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
- It interprets instructions.
- It controls data flow inside the processor.
- It receives external instructions or commands to which it converts to sequence of control signals.
- It controls many execution units(i.e. ALU, data buffers and registers) contained within a CPU.
- It also handles multiple tasks, such as fetching, decoding, execution handling and storing results.

BASIC DEFINATIONS

- **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 changes in microprogram and instruction set can be changed or modified is referred as writeable control Memory.

- **Control Word:** the control variables at any given time can be represented by a control word string of 1’s and 0’s called control word.

BASIC DEFINATIONS

- **Microoperations:** In CPU, micro-operations are detailed low level instructions used in some designs to implement complex machine instructions.

- **Micro instructions:** A symbolic microprogram can be translated into binary equivalent by means of an assembler.
- Each line of the assembly language microprogram defines a symbolic microinstruction.

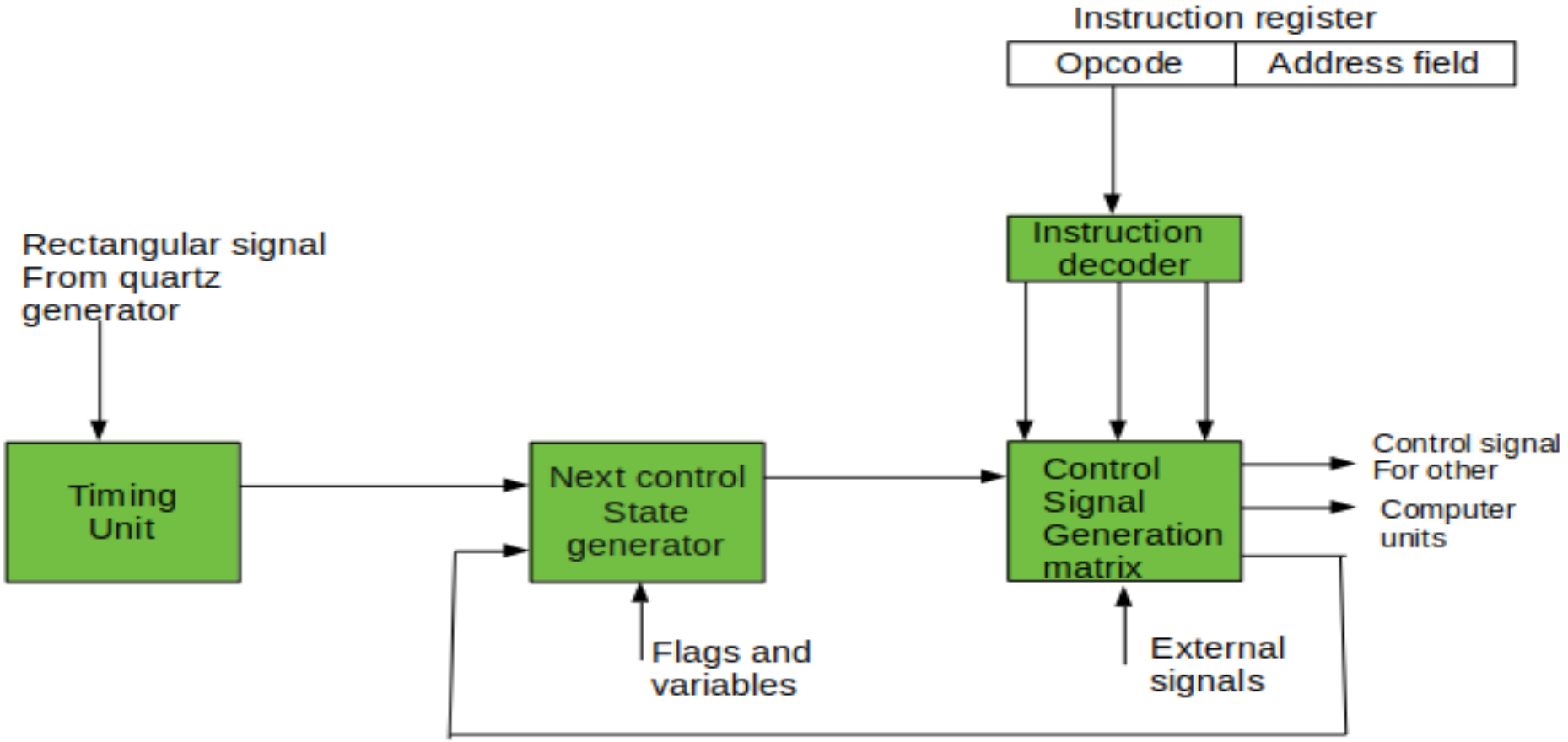
- **Micro program :** A sequence of microinstructions constitutes a micro program.

Types of Control Unit

- There are two types of control units:
- Hardwired control unit
- Microprogrammed control unit.

Hardwired control unit

- When the Control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired.
- It is implemented with the help of gates, flip flops, decoders etc. in the hardware. The inputs to control unit are the instruction register, flags, timing signals etc. This organization can be very complicated if we have to make the control unit large.
- If the design has to be modified or changed, all the combinational circuits have to be modified which is a very difficult task.



Block diagram of a hardwired control unit of a computer

- The operation code of an instruction contains the basic data for control signal generation. In the instruction decoder, the operation code is decoded. The instruction decoder constitutes a set of many decoders that decode different fields of the instruction opcode.
- So, few output lines going out from the instruction decoder obtains active signal values. These output lines are connected to the inputs of the matrix that generates control signals for executive units of the computer. This matrix implements logical combinations of the decoded signals from the instruction opcode with the outputs from the matrix that generates signals representing consecutive control unit states and with signals coming from the outside of the processor, e.g. interrupt signals.

- Control signals for an instruction execution have to be generated not in a single time point but during the entire time interval that corresponds to the instruction execution cycle. Following the structure of this cycle, the suitable sequence of internal states is organized in the control unit.

- A number of signals generated by the control signal generator matrix are sent back to inputs of the next control state generator matrix. This matrix combines these signals with the timing signals, which are generated by the timing unit based on the rectangular patterns usually supplied by the quartz generator.

- When a new instruction arrives at the control unit, the control units is in the initial state of new instruction fetching. Instruction decoding allows the control unit enters the first state relating execution of the new instruction, which lasts as long as the timing signals and other input signals as flags and state information of the computer remain unaltered.
- A change of any of the earlier mentioned signals stimulates the change of the control unit state.
- This causes that a new respective input is generated for the control signal generator matrix.

- When an external signal appears, (e.g. an interrupt) the control unit takes entry into a next control state that is the state concerned with the reaction to this external signal (e.g. interrupt processing). The values of flags and state variables of the computer are used to select suitable states for the instruction execution cycle.
- The last states in the cycle are control states that commence fetching the next instruction of the program: sending the program counter content to the main memory address buffer register and next, reading the instruction word to the instruction register of computer.
- When the ongoing instruction is the stop instruction that ends program execution, the control unit enters an operating system state, in which it waits for a next user directive.

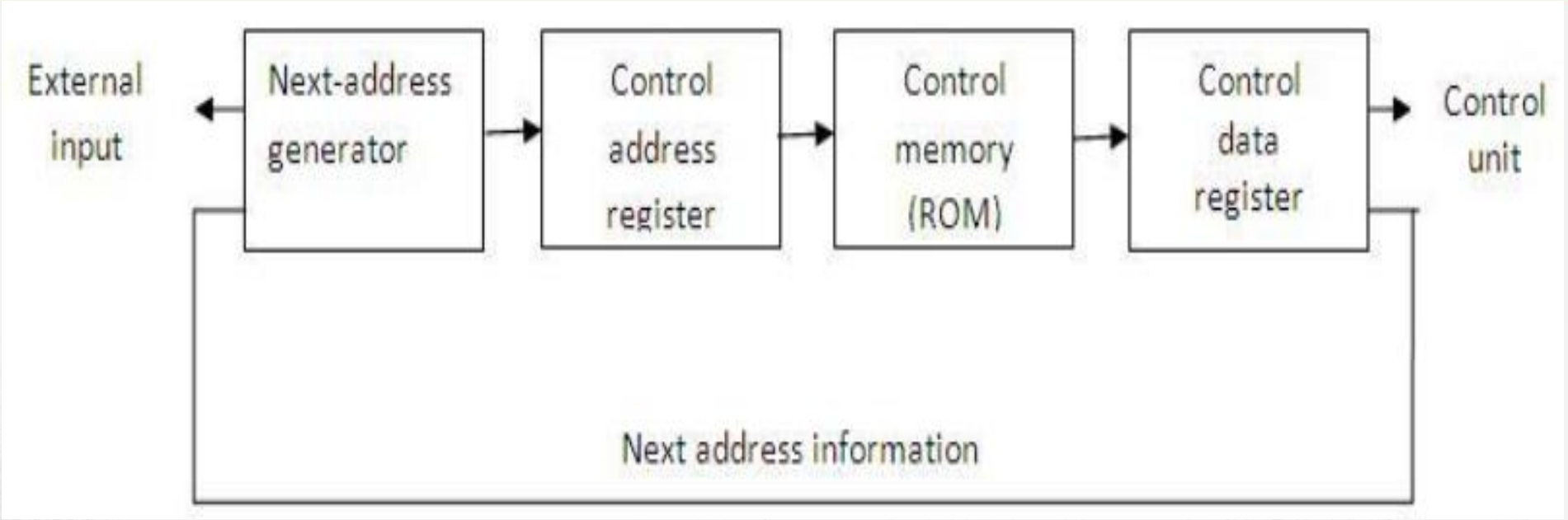
- Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
- Hardwired control is faster than micro-programmed control.
- A controller that uses this approach can operate at high speed.
- RISC architecture is based on hardwired control unit

- ADV:
- Fast control signal generation (using combinational circuit)
- Performance is high
- DISADV:
- Modification is very difficult
- Difficult to correct mistakes or adding new features in existing design

Microprogrammed Control Unit

Microprogrammed control unit

- A control unit whose binary control variables are stored in memory is called a macro programmed control unit.
- It is implemented by using programming approach. A sequence of micro operations is carried out by executing a program consisting of micro-instructions. In this organization any modifications or changes can be done by updating the micro program in the control memory by the programmer.



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

- 1) Control Memory
  - A memory is part of a control unit : *Microprogram*
  - Computer Memory (*employs a microprogrammed control unit*)
    - Main Memory : for storing user program (*Machine instruction/data*)
    - Control Memory : for storing microprogram (*Microinstruction*)
- 2) Control Address Register
  - Specify the address of the microinstruction
- 3) Sequencer (= *Next Address Generator*)
  - Determine the address sequence that is read from control memory
  - Next address of the next microinstruction can be specified several way depending on the sequencer input :
- 4) Control Data Register (= *Pipeline Register*)
  - Hold the microinstruction read from control memory
  - Allows the execution of the microoperations specified by the control word *simultaneously* with the generation of the next microinstruction

- The Control memory address register specifies the address of the micro-instruction.
- The Control memory is assumed to be a ROM, within which all control information is permanently stored.
- The control register holds the microinstruction fetched from the memory.
- The micro-instruction contains a control word that specifies one or more micro-operations for the data processor.
- While the micro-operations 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.
- The next address generator is often referred to as a micro-program sequencer, as it determines the address sequence that is read from control memory.

- The control data register holds the present microinstruction while the next address is computed and read from memory.
- 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.
- The main advantage of the micro programmed control is the fact that once the hardware configuration is established; there should be no need for further hardware or wiring changes.
- If we want to establish a different control sequence for the system, all we need to do is specify a different set of microinstructions for control memory

Difference between Hardwired Control and Microprogrammed Control

Hardwired Control	Microprogrammed Control
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.
Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.

Address Sequencing

- Microinstructions are stored in control memory in groups, with each group specifying a ***routine***.
- To appreciate the address sequencing in a micro-program control unit, let us specify the steps that the control must undergo during the execution of a single computer instruction.

Steps of Address Sequencing

- **Step-1:**
  An initial address is loaded into the control address register when power is turned on in the computer.

- This address is usually the address of the first microinstruction that activates the instruction fetch routine.

- The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions.

- At the end of the fetch routine, the instruction is in the instruction register of the computer.

- **Step-2:**
  The control memory next must go through the routine that determines the effective address of the operand.

- A machine instruction may have bits that specify various addressing modes, such as indirect address and index registers.

- The effective address computation routine in control memory can be reached through a branch microinstruction, which is conditioned on the status of the mode bits of the instruction.

- When the effective address computation routine is completed, the address of the operand is available in the memory address register.

- **Step-3:**
  The next step is to generate the microoperations that execute the instruction fetched from memory.

- The microoperation steps to be generated in processor registers depend on the operation code part of the instruction.

- Each instruction has its own micro-program routine stored in a given location of control memory.

- The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a ***mapping*** process.

- A mapping procedure is a rule that transforms the instruction code into a control memory address.

- **Step-4:**
  Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.

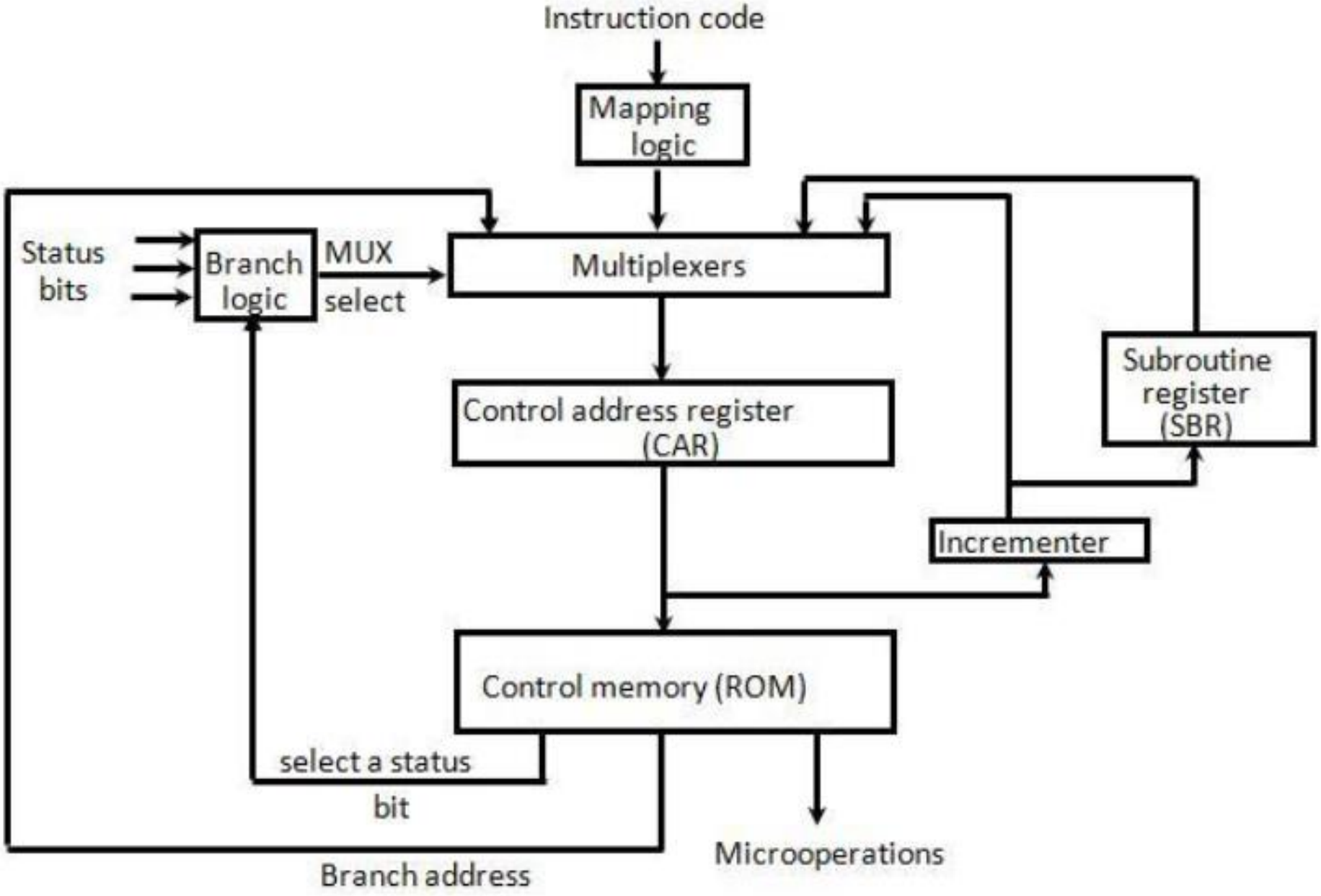
- Micro-programs that employ subroutines will require an external register for storing the return address.

- Return addresses cannot be stored in ROM because the unit has no writing capability.

- When the execution of the instruction is completed, control must return to the fetch routine.

- This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine

selection of address for control memory



- Multiplexer
  - ① CAR Increment
  - ② JMP/CALL
  - ③ Mapping
  - ④ Subroutine Return
- CAR : Control Address Register
  - CAR receive the address from 4 different paths
    1) Incrementer
    2) Branch address from control memory
    3) Mapping Logic
    4) SBR : Subroutine Register
  - Return Address for a subroutine is stored in SBR

- SBR : Subroutine Register
  - Return Address can not be stored in ROM

- Conditional Branching
  - Status Bits
    - Control the conditional branch decisions generated in the ***Branch Logic***
  - Branch Logic
    - Test the specified condition and Branch to the indicated address if the condition is met ; otherwise, the control address register is just incremented.

- The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers and other bits to specify the method by which the next address is obtained.

- 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.

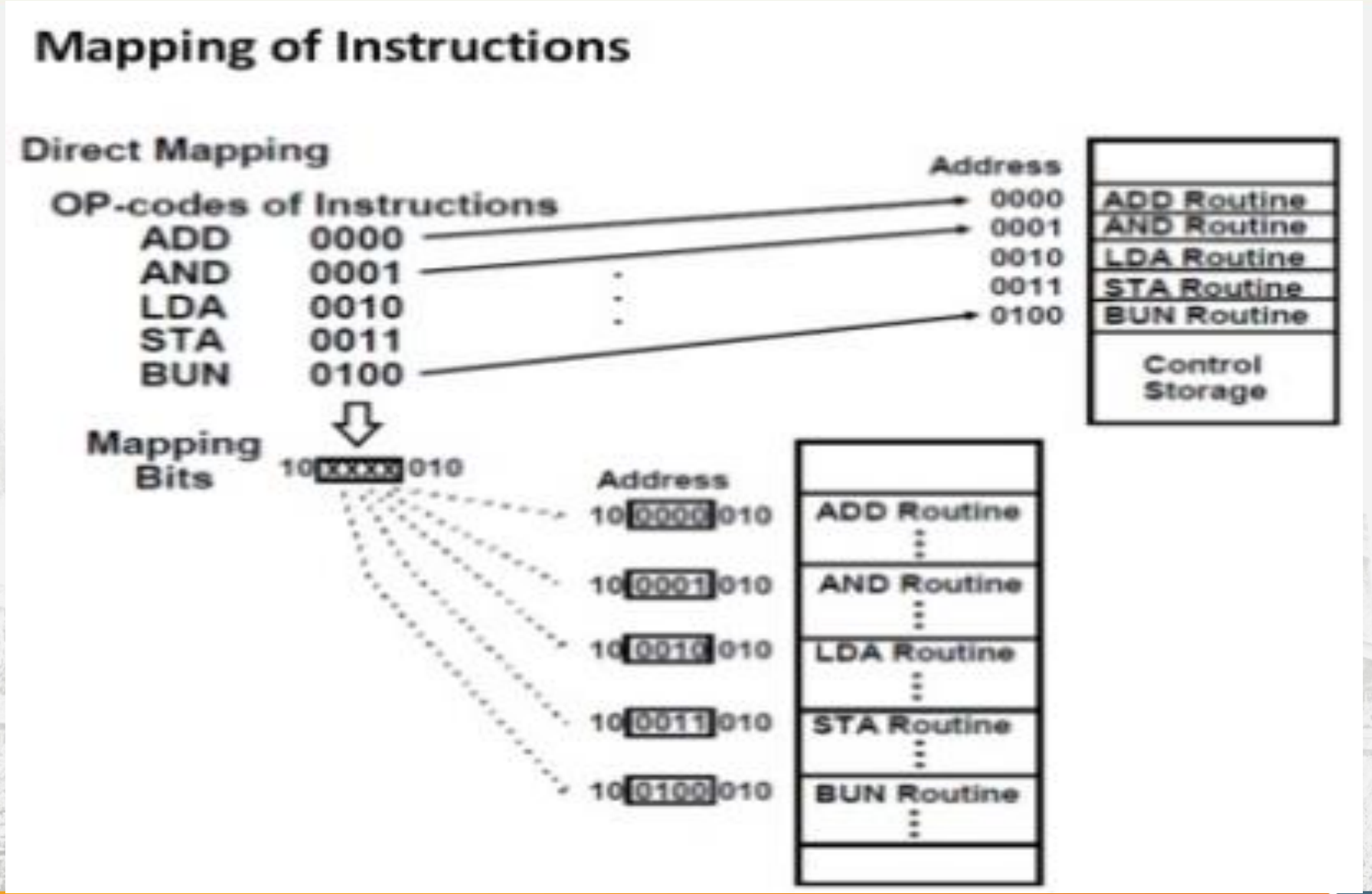
- Branching is achieved by specifying the branch address in one of the fields of the microinstruction.
- Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition.
- An external address is transferred into control memory via a mapping logic circuit.
- The return address for a subroutine is stored in a special register whose value is then used when the micro-program wishes to return from the subroutine.

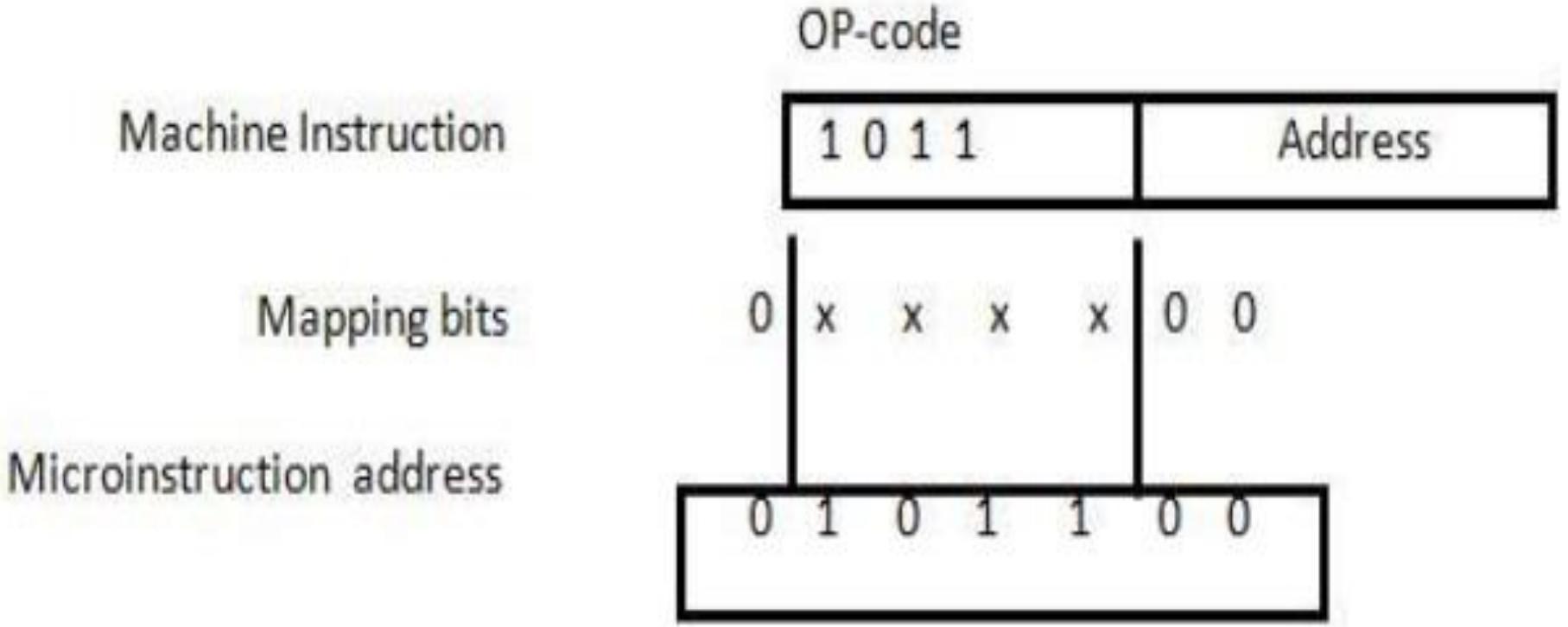
- The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and input or output status conditions.

- The status bits, together with the field in the microinstruction that specifies a branch address, control the conditional branch decisions
- generated in the branch logic.

- A 1 output in the multiplexer generates a control signal to transfer the branch address from the microinstruction into the control address register.
  A 0 output in the multiplexer causes the address register to be incremented.

Mapping of Instruction



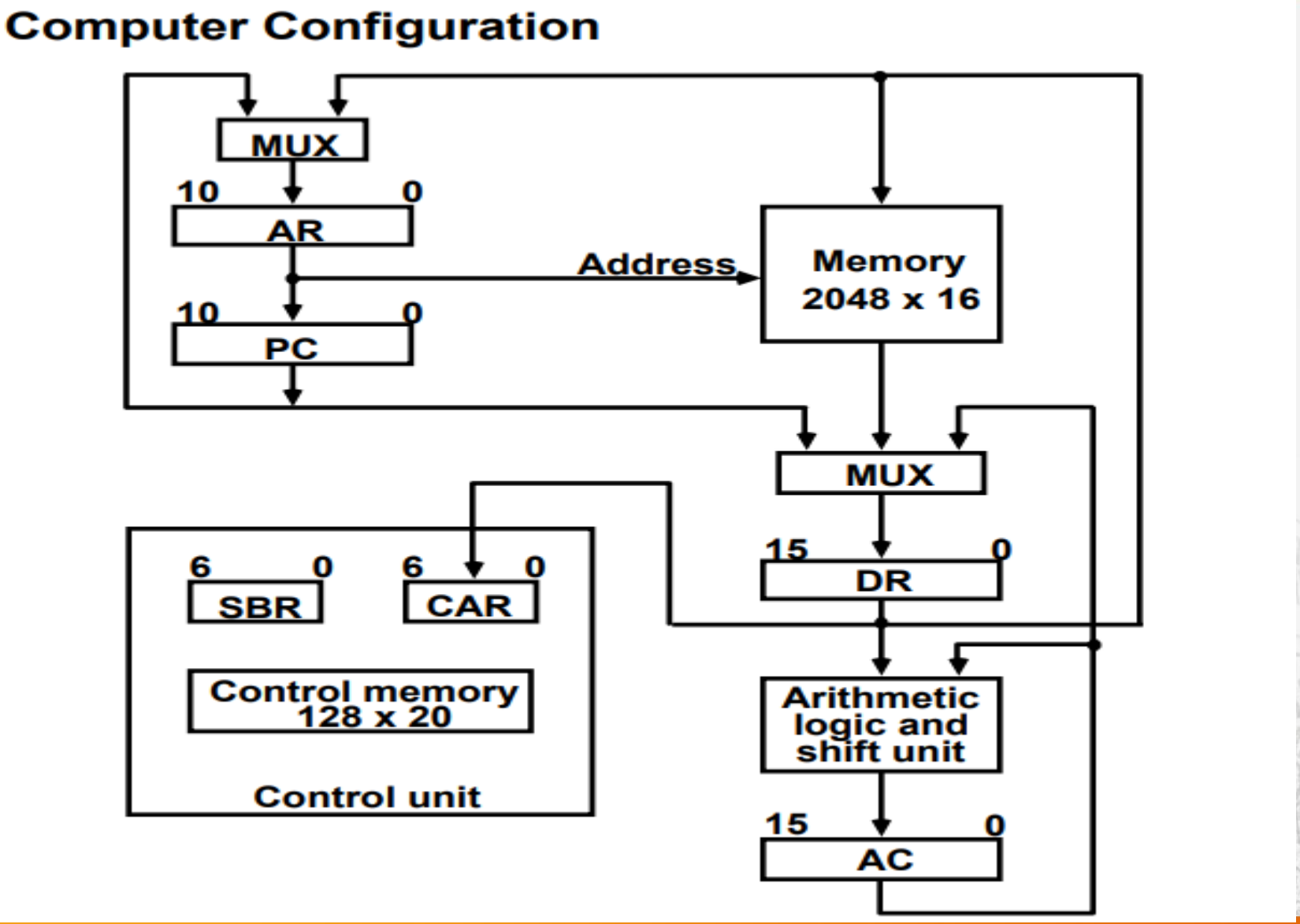

OP-code
Machine Instruction
1 0 1 1
Address
Mapping bits
0 x x x x 0 0
Microinstruction address
0 1 0 1 1 0 0

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

- 4 bit Opcode = specify up to 16 distinct instruction
- Mapping Process : Converts *the 4-bit Opcode* to *a 7-bit control memory address*
  - 1) Place a “0” in the most significant bit of the address
  - 2) Transfer 4-bit Operation code bits
  - 3) Clear the two least significant bits of the CAR Mapping Function : Implemented by *Mapping ROM* or *PLD*
- Control Memory Size : 128 words
- In this way the microprogram routine that executes the instruction can be placed in any desired location in control memory.
- The mapping concept provides flexibility for adding instructions for control memory as the need arises

Microprogram example

- The microcode generation is called microprogramming and is a process similar to conventional machine language programming

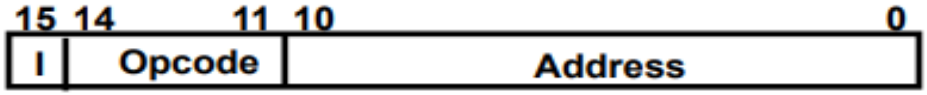

Computer Configuration
MUX
10
0
AR
Address
Memory
2048 x 16
10
0
PC
MUX
15
0
DR
6
0
6
0
SBR
CAR
Control memory
128 x 20
Control unit
Arithmetic
logic and
shift unit
15
0
AC

- The block diagram of the computer is shown in Figure 4.4. It consists of
  1. Two memory units:
  Main memory -> for storing instructions and data, and
  Control memory -> for storing the microprogram.
  2. Six Registers:
  Processor unit register: AC(accumulator),PC(Program Counter),
  AR(Address Register),
  DR(Data Register)
  Control unit register: CAR (Control Address Register), SBR(Subroutine Register)

- 3. Multiplexers:
  The transfer of information among the registers in the processor is done through
  multiplexers rather than a common bus.
  4. ALU:
  The arithmetic, logic, and shift unit performs microoperations with data from AC and DR
  and places the result in AC

- DR can receive information from AC, PC, or memory.
  AR can receive information from PC or DR.
  PC can receive information only from AR.
  Input data written to memory come from DR, and data read from memory can go only to DR
  .

Machine instruction format

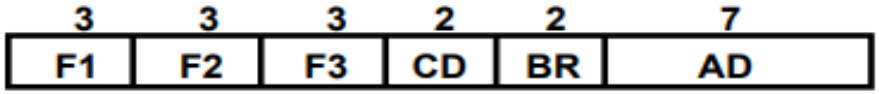


Sample machine instructions

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



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 ← 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 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 ← 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}

a. Symbolic Address : Label (= Address)

b. Symbol “NEXT” : next address

c. Symbol “RET” or “MAP” : AD field = 0000000

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

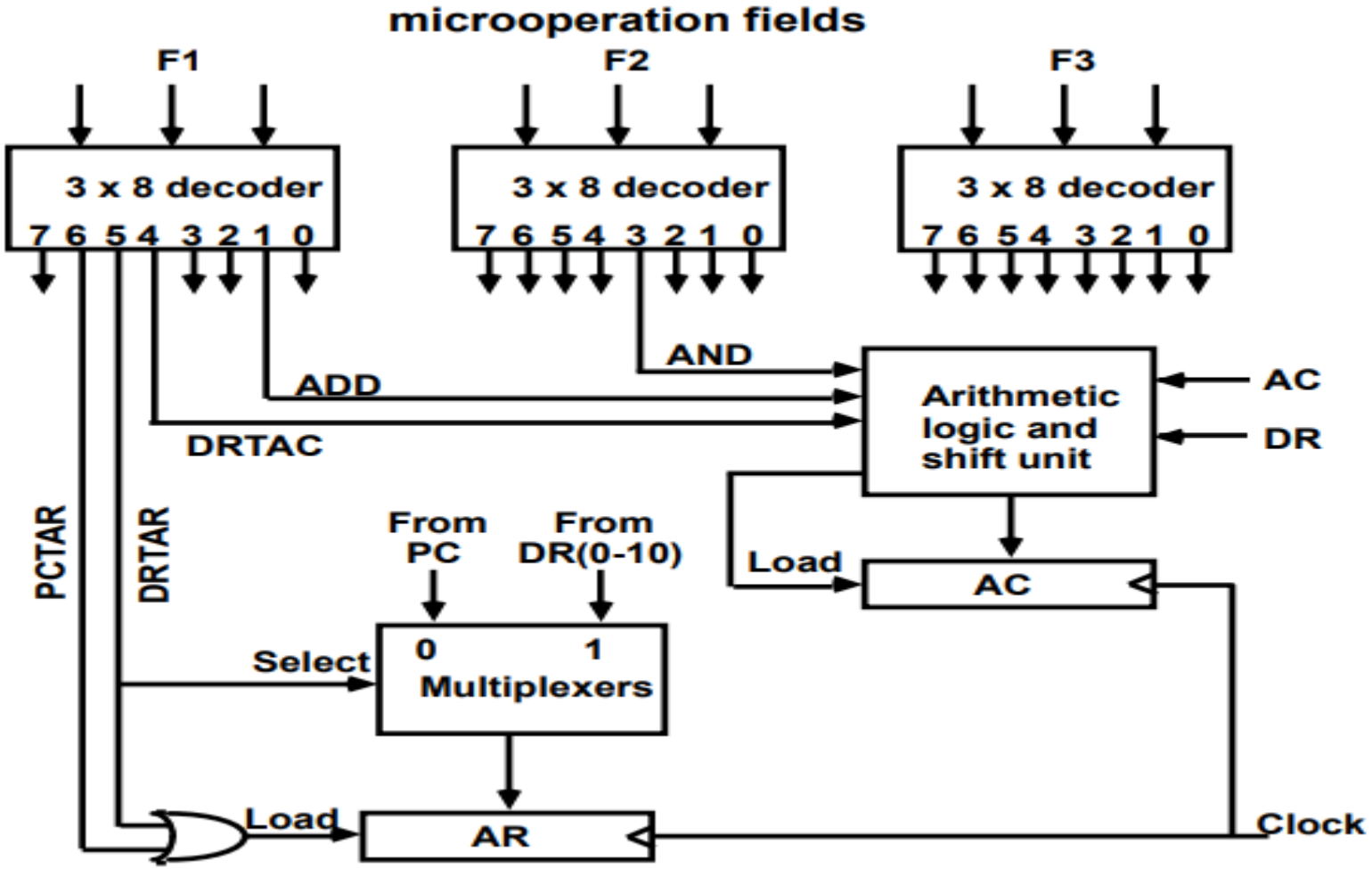
Label	Microops	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	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

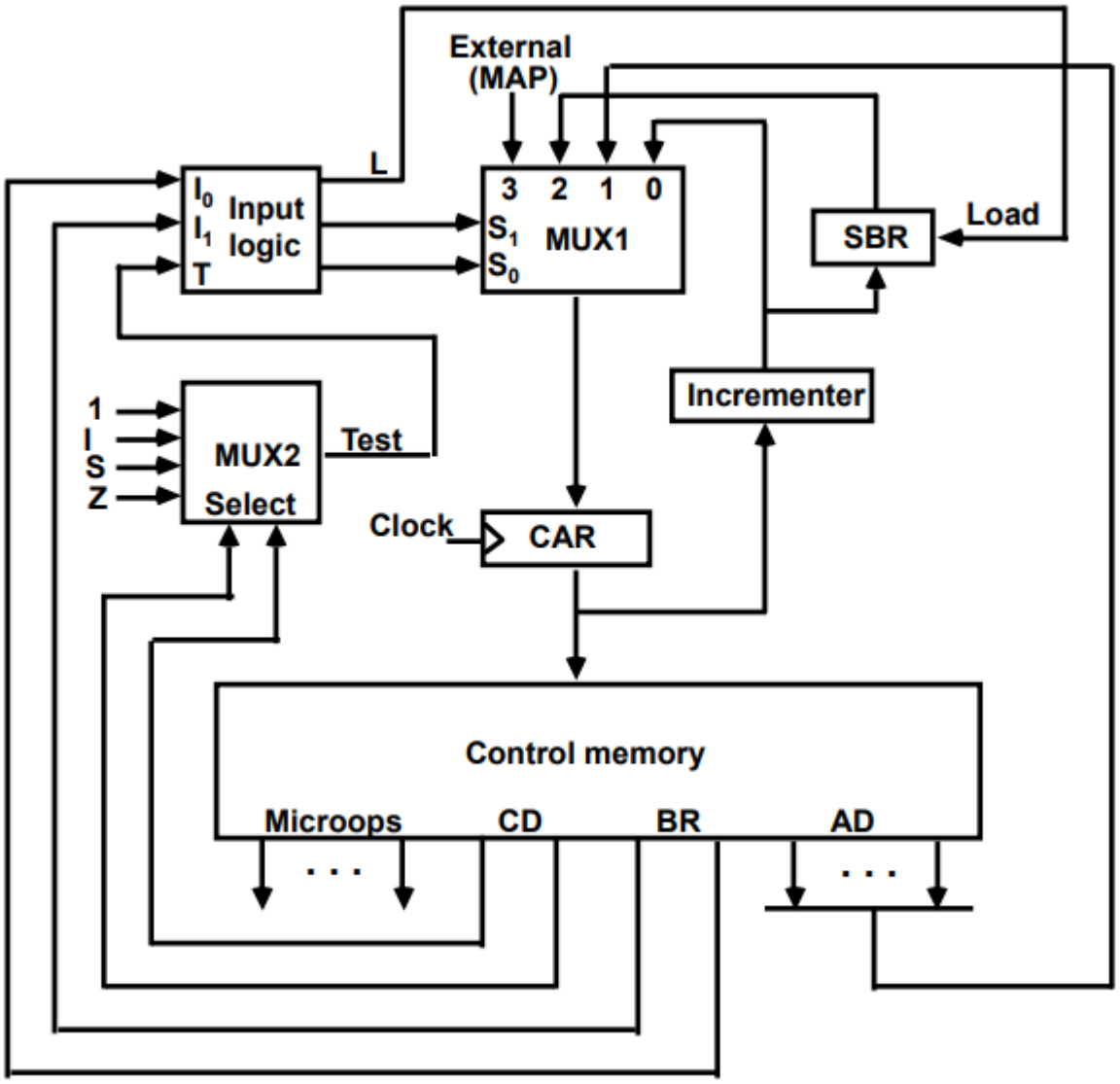


- PCTAR- Transfer from **PC** to **AR,** DRTAC- Transfer from **DR to AC, DRTAC-** Transfer from **DR** to **AC**

- Decoding of Microinstruction Fields :
  - F1, F2, and F3 of Microinstruction are decoded with a 3 x 8 decoder
  - Output of decoder must be connected to the proper circuit to initiate the corresponding microoperation
    - F1 = 101 (5) : **DRTAR**

      F1 = 110 (6) : **PCTAR**
      - Output 5 and 6 of decoder F1 are connected to the load input of AR (*two input of OR gate*)
      - Multiplexer select the data from DR when output 5 is active
      - Multiplexer select the data from AC when output 5 is inactive
  - Arithmetic Logic Shift Unit
    - Control signal of ALU in *hardwired control* :
    - Control signal will be now come from the *output of the decoders* associated with the AND, ADD, and DRTAC.

MICROPROGRAM SEQUENCER



- The basic components of a microprogrammed control unit are the control memory and the circuits that select the next address.
  The address selection part is called a microprogram sequencer.
  A microprogram sequencer can be constructed with digital functions to suit a particular application.
  To guarantee a wide range of acceptability, an integrated circuit sequencer must provide an internal organization that can be adapted to a wide range of applications.
  The purpose of a microprogram sequencer is to present an address to the control memory
  so that a microinstruction may be read and executed.

- Commercial sequencers include within the unit an internal register stack used for temporary storage of addresses during microprogram looping and subroutine calls.
  Some sequencers provide an output register which can function as the address register for the control memory.

  There are two multiplexers in the circuit.
- The first multiplexer selects an address from one of four sources and routes it into a control address register CAR.

- The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic circuit.

- The output from CAR provides the address for the control memory.
- The content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine registers SBR.
- The other three inputs to multiplexer 1 come from the address field of the present microinstruction, from the output of SBR, and from an external source that maps the instruction.
- A single subroutine register, a typical sequencer will have a register stack about four to eight levels deep. In this way, a number of subroutines can be active at the same time.

- The CD (condition) field of the microinstruction selects one of the status bits in the second multiplexer.
- If the bit selected is equal to 1, the T (test) variable is equal to 1; otherwise, it is equal to 0.
- The T value together with the two bits from the BR (branch) field goes to an input logic circuit.
- The input logic in a particular sequencer will determine the type of operations that are available in the unit.

Input Logic : Truth Table

BR	Input			MUX 1		Load SBR
	I1	I0	T	S1	S0	L
0 0	0	0	0	0	0	0
0 0	0	0	1	0	1	0
0 1	0	1	0	0	0	0
0 1	0	1	1	0	1	1
1 0	1	0	X	1	0	0
1 1	1	1	X	1	1	0

Memory :

As the word implies “memory” means the place where we have to store data or information.

There are various units which are used to measure computer memory

- Bit -Smallest unit of computer memory
- Byte-8 bit = 1 byte
- Kilobyte-1024 byte = 1 KB
- Megabyte-1024 KB = 1 MB
- Gigabyte-1024 MB = 1 GB
- Terabyte-1024 GB = 1 TB

MEMORY CLASSIFICATION

1) Primary Memory:-

- Primary memory also known as “main memory” or “internal memory” which is located in the motherboard of system or as we say which is directly connected to the CPU. It is the place where only little bit of data are stored either by manufacturer or by user.
- This is further divided into two parts:-
- RAM
- ROM

2) Secondary Memory:-

- The place where we store our personal data in computer system
- It is non-volatile in nature so that we cannot loose the data when power supply is off.
- There are two methods for accessing the data from it:-
- (A) Sequential–
- This is the method in which we search the data sequentially or line by line until you find the desired data. E.g. Magnetic tape,
- (B) Direct–
- This is the method in which computer can go directly to the information that the user wants.
- e.g.magnetic disk,optical disk,etc

Memory Classification

With respect to the way of data access we can classify memories as:

- random access memories (RAM),
- sequentially accessible memory (SAM),
- direct access memory (DAM),
- contents addressable memory (CAM).

Access time - the interval of time between the instant of data read/write request, and the instant at which the delivery of data is completed or its storage is started.

RAM modules

SAM/DAM memories:

HDDs

tape memories

CD-, DVD-ROMS.

CAM memories find applications in:

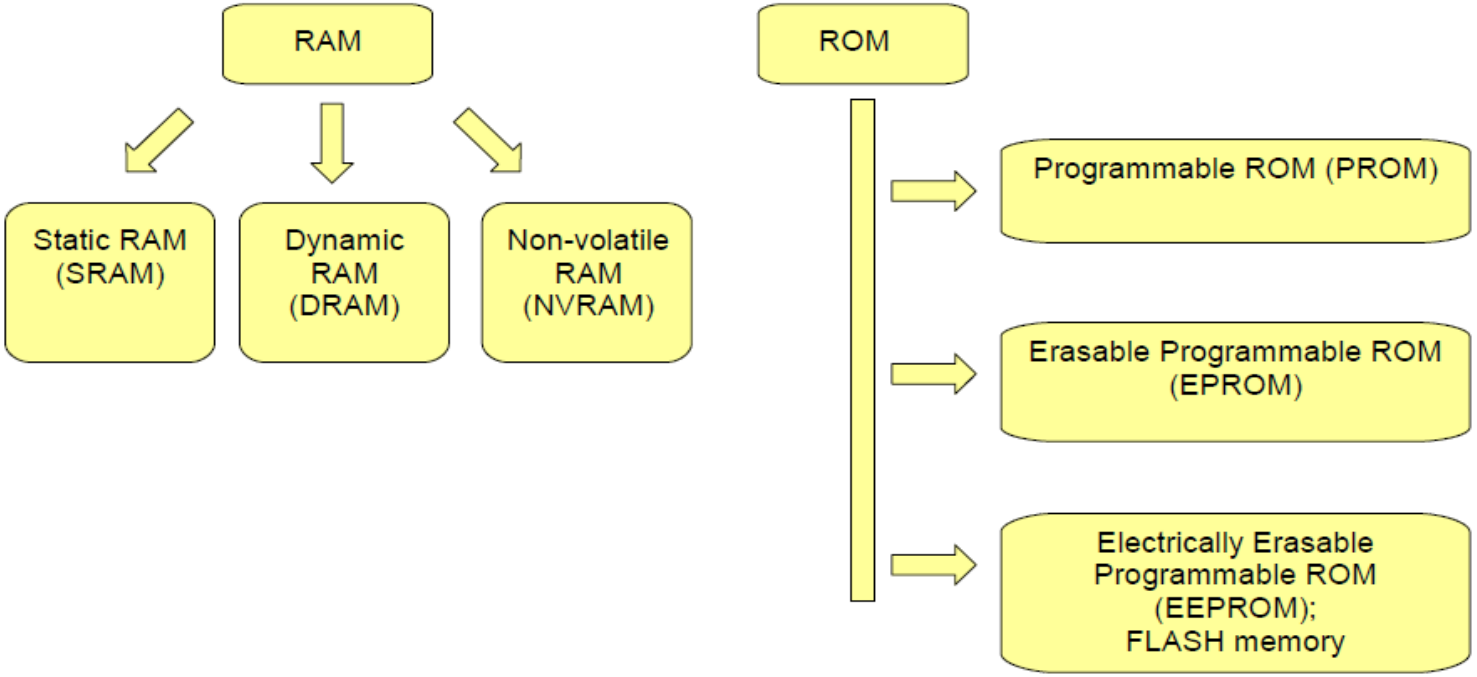
switches, routers etc.

CPUs in cache controllers

Random access memory - the access time to any piece of data is independent to the physical location of data. Access time is constant.

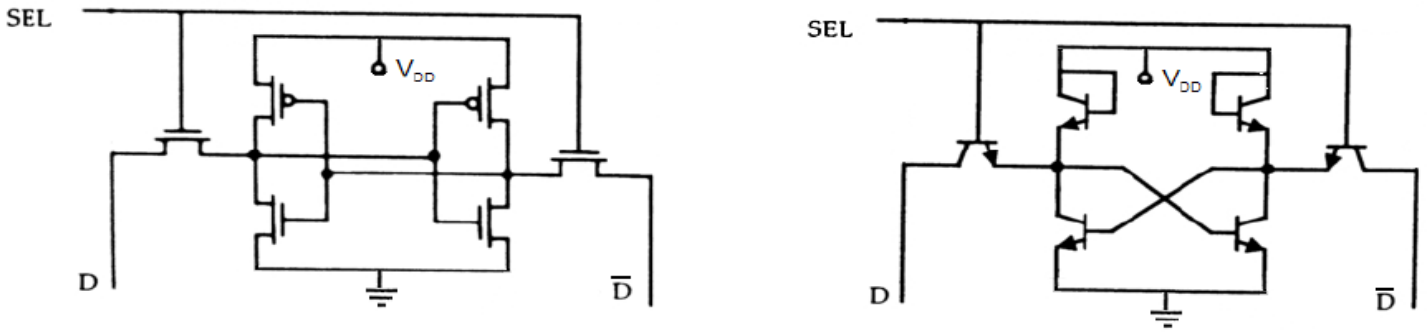
Random access memories can be further classified as:
- read-write memories (usually referred to as RAMs),
- read-only memories (ROM).

Among random access and read-only memories we distinguish:



Random Access Memories

Static random access memories (SRAM) - one-bit memory cells use bistable latches for data storage and hence, unlike for dynamic RAM, there is no need to periodically refresh memory contents.



Schematics of one-bit cell for static random access memory

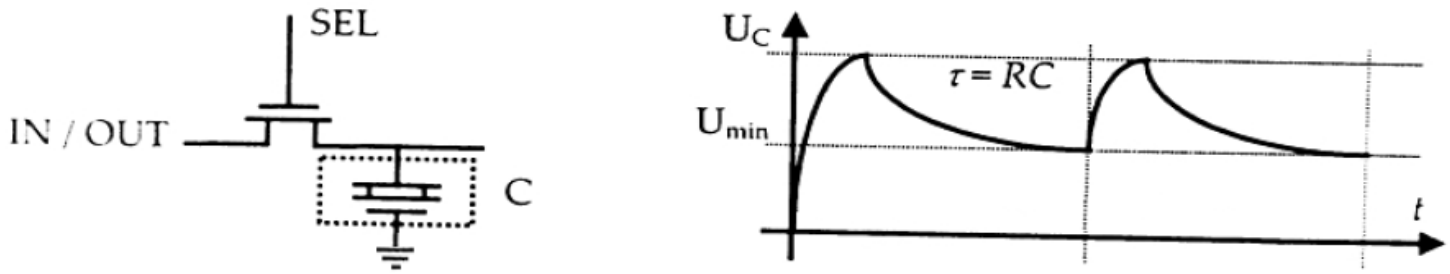
In order to **write** data into SRAM cell it is required to activate line SEL and provide bit of information and its inverse at inputs D and $\overline{D}$ respectively.

Reading operation requires to activate SEL line. The bit of data is available at D line.

Pros and Cons:
- faster and less power hungry than DRAMs,
- more expensive (6 transistors per cell).

Random Access Memories

Dynamic random access memories (DRAM) - each one-bit memory cell uses a capacitor for data storage. Since capacitors leak there is a need to refresh the contents of memory periodically (usually once in τ=0,5 ÷ 2 ms).



Memory cell for dynamic random access memory*

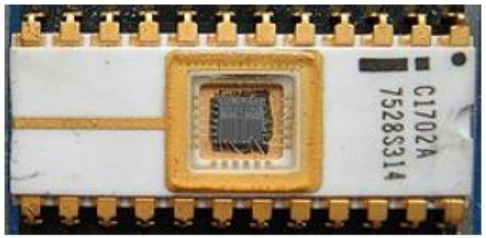
Both **read** and **write** operations require to open the transistor by providing high voltage on line SEL. The bit of data to be written must be given at line IN/OUT. During reading operation the bit of data is available at the same line.

Pros and Cons:
- less expensive (only one transistor per cell),
- slower than SRAM.

Read Only Memories

Programmable read only memories (PROM) - are programmed during manufacturing process. The contents of each memory cell is locked by a fuse or antifuse (diodes). PROMs are used for permanent data storage.

Erasable read only memories (EPROM) - there is a possibility to erase EPROM with ultraviolet light (about 20 minutes) what sets all bits in memory cells to 1. Programming requires higher voltage. Memory cells are built with floating gate transistors. Data can be stored in EPROMs for about 10 years.

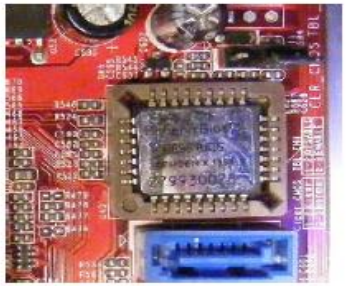


Example of EPROM chip with glass window admitting UV light

Electrically erasable read only memories (EEPROM) - erasing does not require ultraviolet light but higher voltage and can be applied not to the whole circuit but to each memory cell separately.

Other Types of RAMs

Non-volatile random access memory (NVRAM) - with this name we refer generally to any memory which does not lose information when power is turned off. Except from ROMs, NVRAM also include conventional volatile random access memories with battery backup such as BIOS memory (Basic Input Output System).



BIOS memory chip with battery*

Flash memory (FLASH) - by this name the cheaper variant of EEPROM is described. In case of FLASH memory not separate bytes but blocks of bytes are being erased at the same time. It makes the construction of such memories cheaper in comparison to regular EEPROMs.

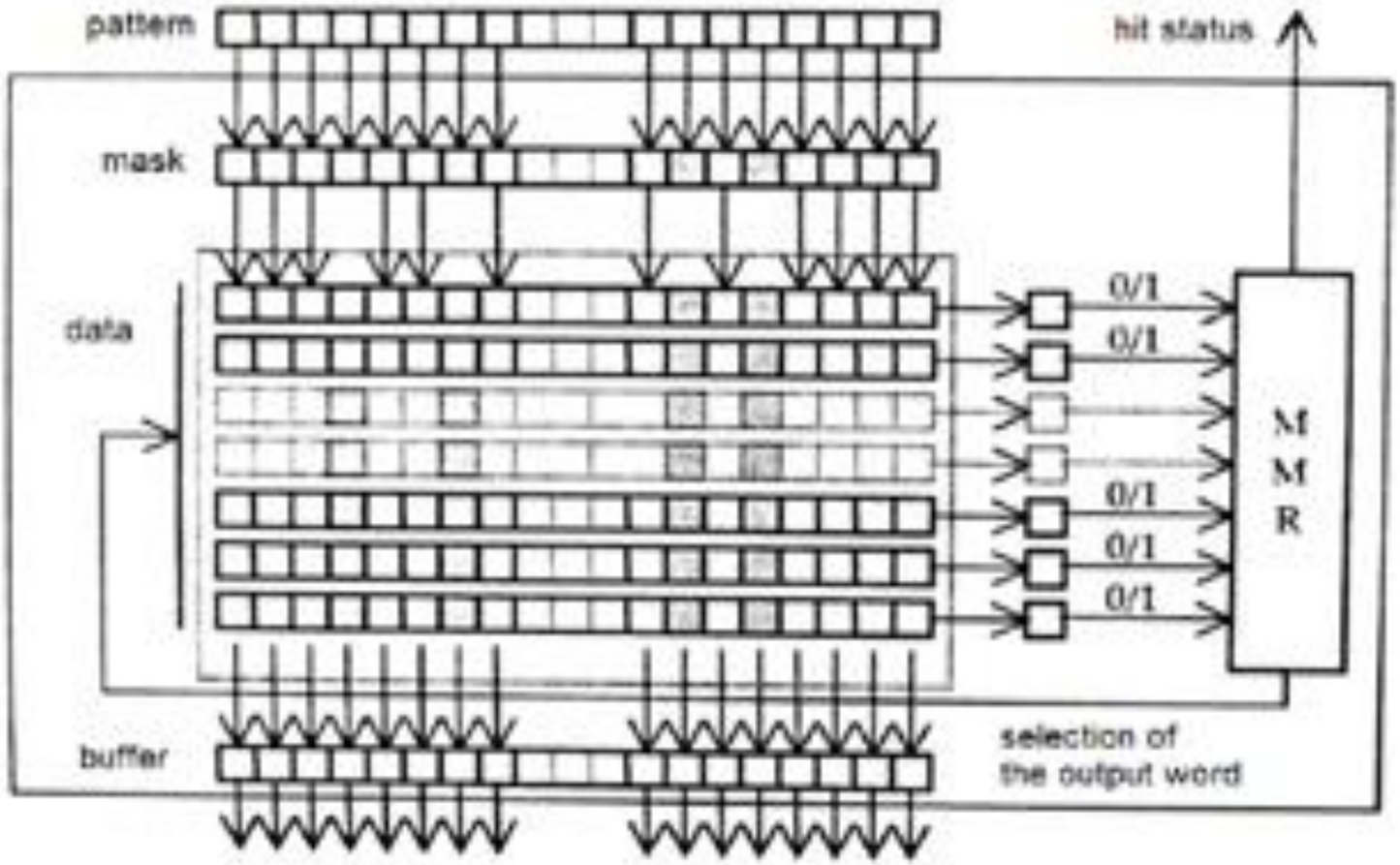
Exemplary applications of FLASH memories: pendrives, memory cards

Associative/CAM (Content Addressable Memory)

- An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.
- - Accessed by the content of the data rather than by an address
- Also called Content Addressable Memory (CAM)

- When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.

- On the other hand, when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified content are located by the memory and are marked for reading.



Logical structure of an associative memory*

CAM/SAM

Applications of CAMs:
- network routing and switching devices where fast resolving of data recipients` addresses is required,
- CPU and disk drive cache memories.

Sequentially addressable memories (SAM) - access to data is realised in a sequential fashion. Fully sequential are buffers organised as FIFO, LIFO, etc. queues. They find their applications in devices controllers, microprocessors, etc.

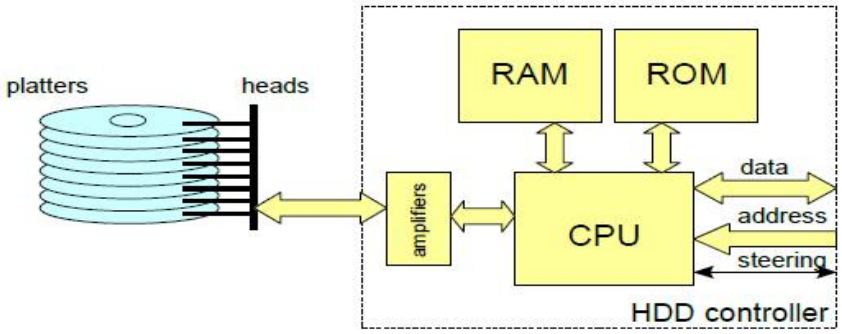
Direct access memory (DAM) - in literature usually by this term authors refer to memories with block organised data. Access to blocks is direct. However data within blocks is accessed sequentially. As examples we can indicate: magnetic and optical disk drives, tape memories, etc.

Sequentially Accessible Memory

Hard disk drive (HDD) - is a kind of mechanical device memory where data is encoded in the form of magnetic impulses on platters covered with magnetising ferromagnetic material.

Hard disk drive memory

The typical HDD consists of: stepper and linear motors, read-and-write heads, platters and disk controller. The controller includes central processing unit, RAM and ROM memories and data amplifiers circuits. Communication between CPU and HDD requires transmission of data, commands (to appropriate registers of HDD controller) and status words.



Sequentially Accessible Memory

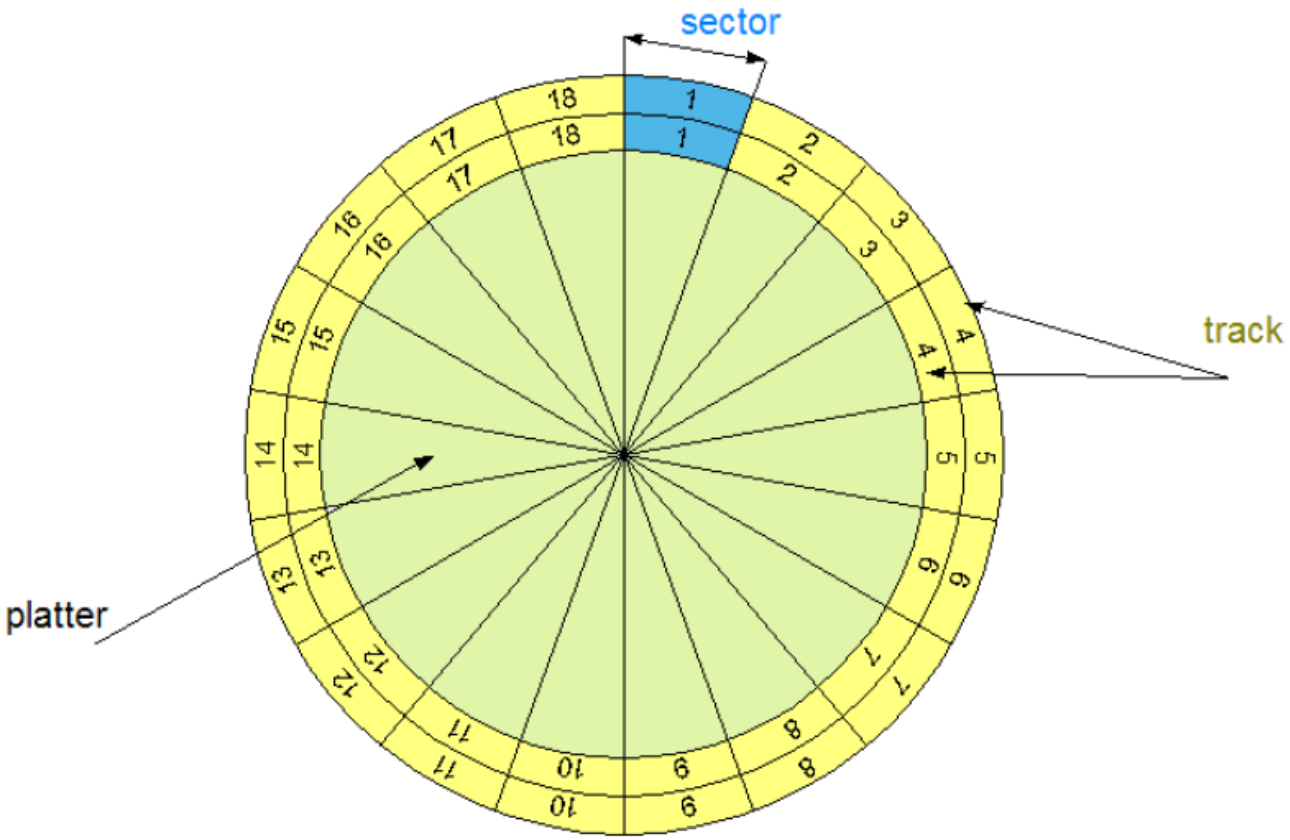
Physical organisation of disks:

head - corresponds with one side of a platter;

track - circular area on platter where data is stored;

sector - a fragment of track that is the smallest portion of data to be read or written on disk;

cylinder - a set of tracks with the same number belonging to different platters.



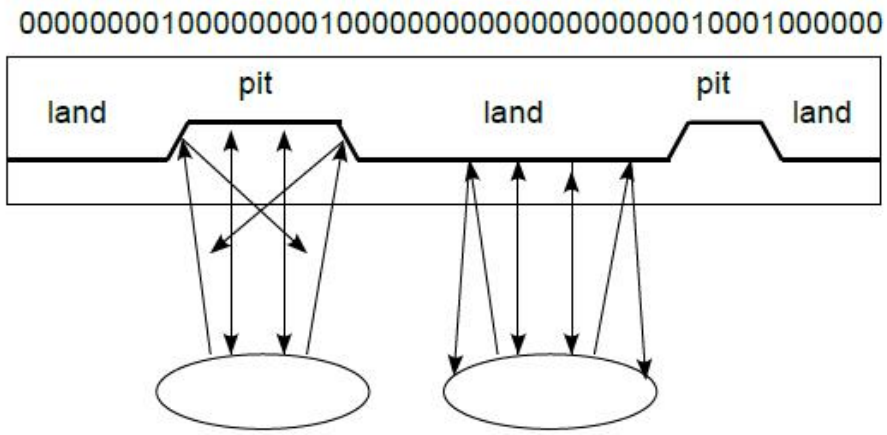
Physical organisation of a disk

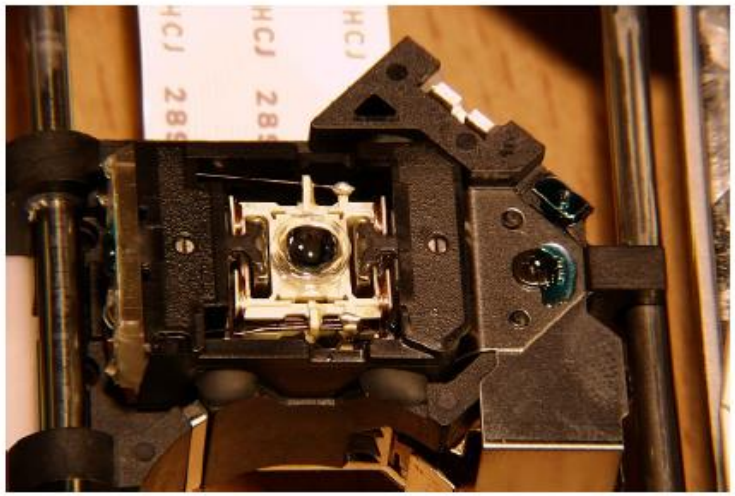
Sequentially Accessible Memory

CD-ROMs

In compact disks for data storage one spiral track is used. Data is stored on the disc as the series of microscopic indentations (“pits”) that cause destructive interference of laser light resulting in reduction of intensity of the reflected beam.

The linear data density is constant what means that the linear rotational speed of the disk is linear. Such approach requires more complicated reading/writing circuits.





Laser structure*