UNIT -3
Micro Programmed Control

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***.***

Control Word:

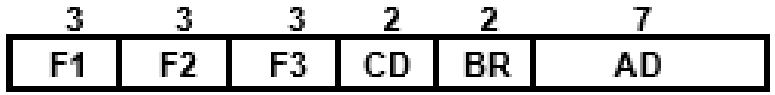
- The control variables at any given time can be represented by a control word string of 1 's and 0's called a control word.

Microoperations:

- In computer central processing units, micro-operations (also known as a microops 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.



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

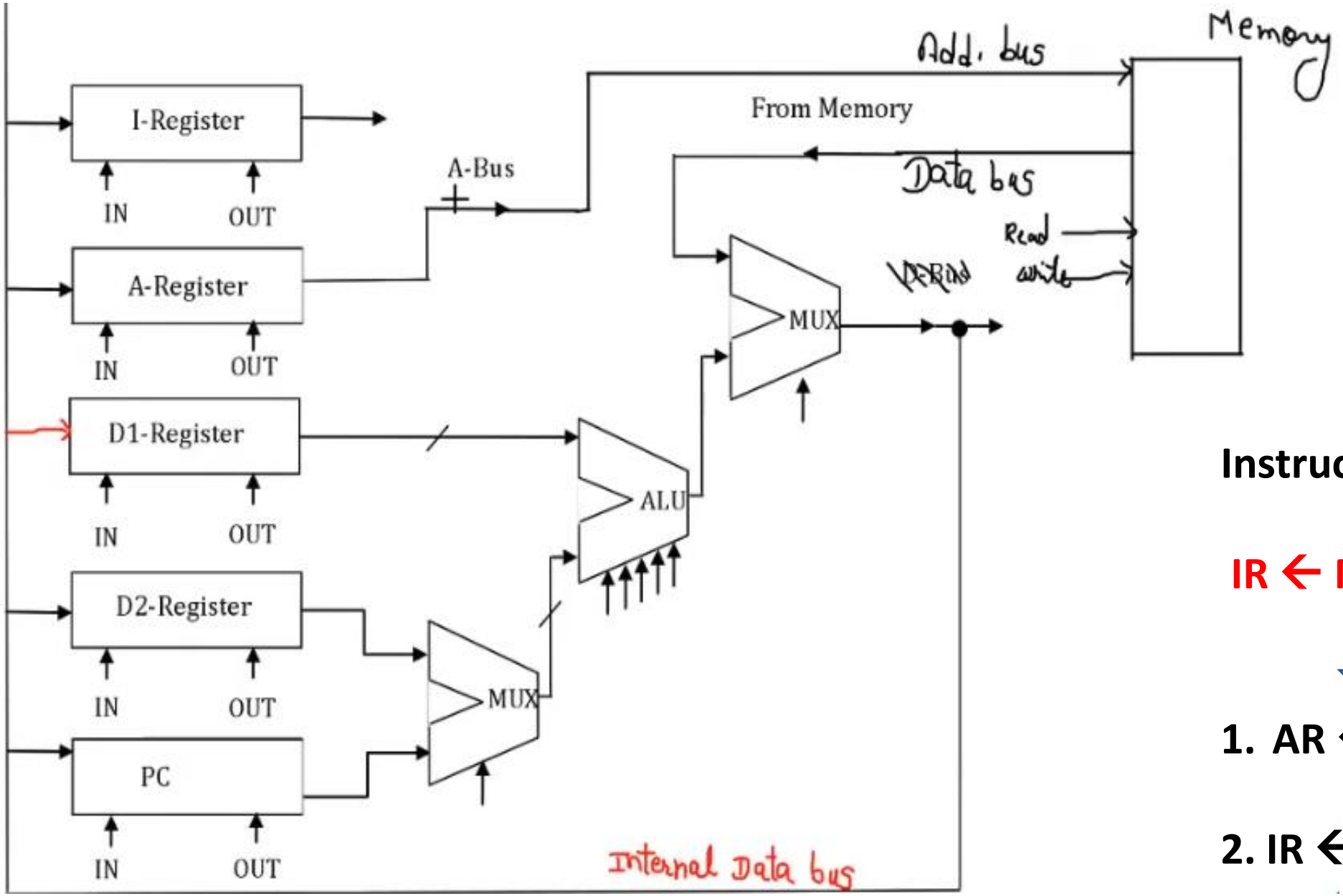
Micro program:

- 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.

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.

Data Path :

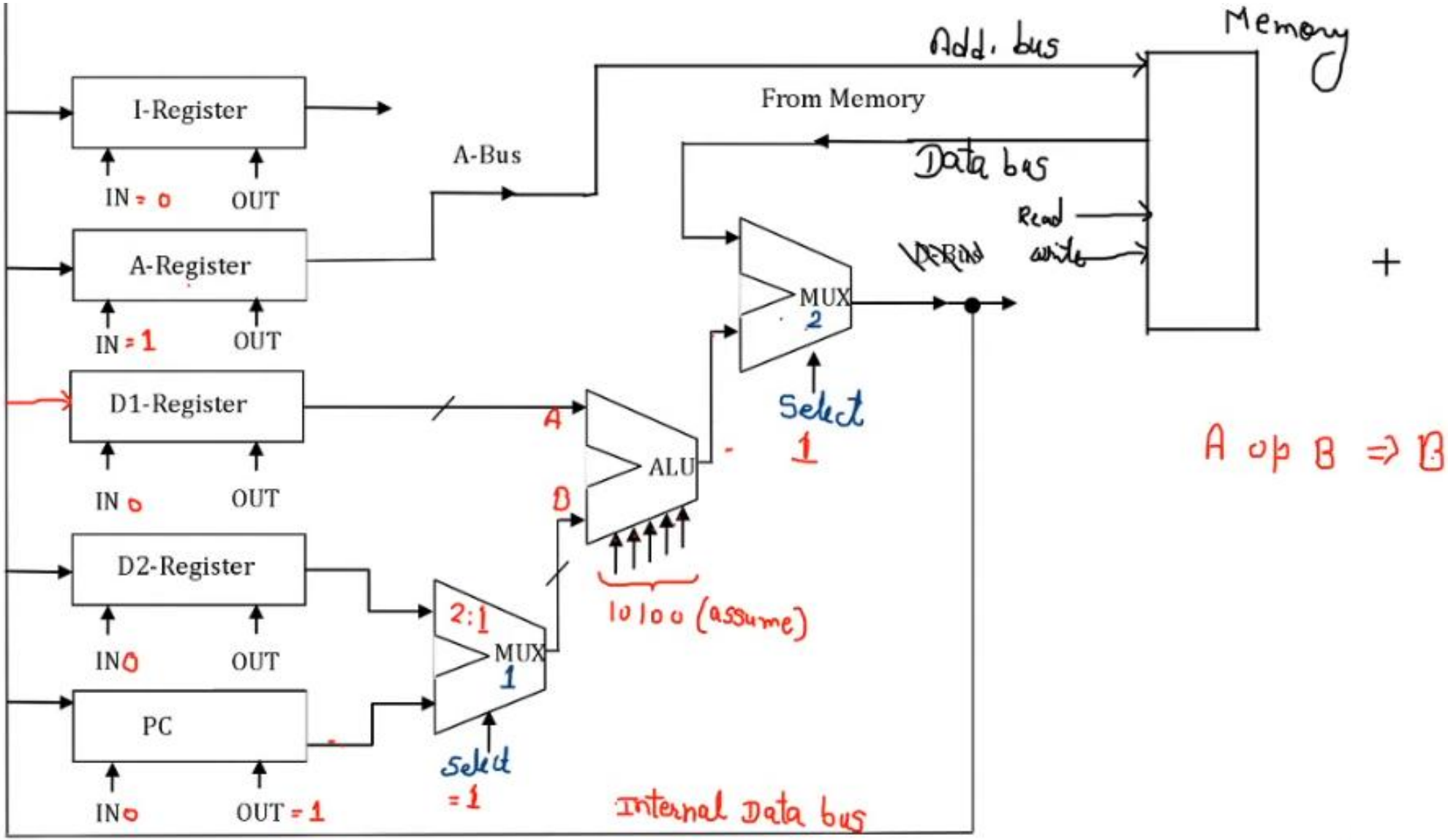


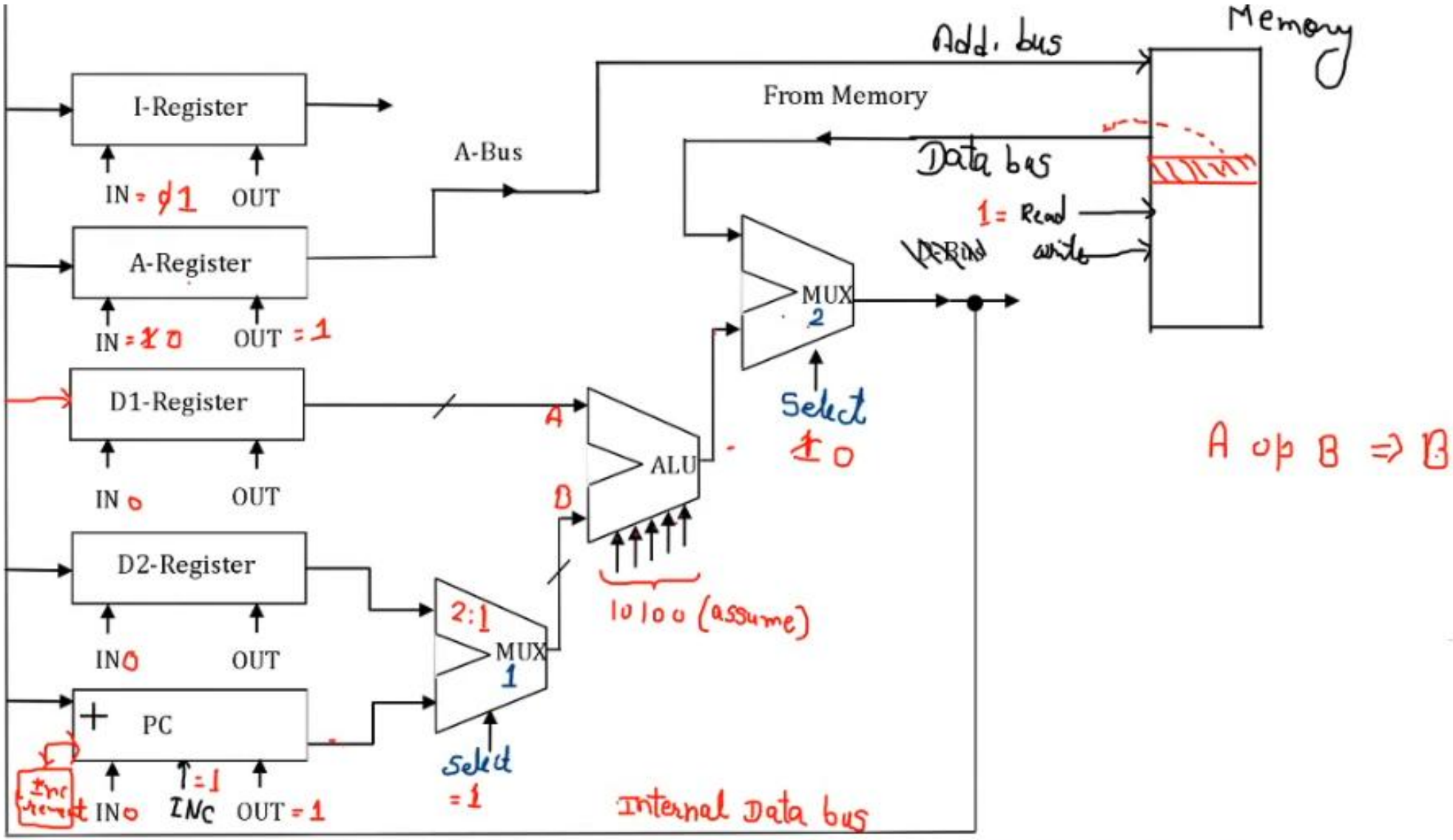
Instruction Fetch:

IR ← M[PC]

1. AR ← PC

2. IR ← M[AR] , PC ← PC+ 1


I-Register
IN = 0
OUT
A-Register
IN = 1
OUT
D1-Register
IN 0
OUT
D2-Register
IN 0
OUT
PC
IN 0
OUT = 1
A-Bus
Add. bus
Memory
From Memory
Data bus
Read
write
MUX 2
Select 1
A
B
ALU
2:1 MUX 1
Select = 1
10100 (assume)
Internal Data bus
A op B ⇒ B


I-Register
IN = 0/1
OUT
A-Register
IN = 1/0
OUT = 1
D1-Register
IN 0
OUT
D2-Register
IN 0
OUT
PC
IN 0
INC
=1
OUT = 1
Inc request
A-Bus
Add. bus
Memory
From Memory
Data bus
1 = Read
0 = Write
MUX 2
Select = 1/0
A
B
ALU
A op B => B
10100 (assume)
2:1 MUX 1
Select = 1
Internal Data bus

Control Unit

generates control signals, sends those to components, components work accordingly.

→ Control variable:- name of control signal

→ Control word:- Collection of all control signals.

DataPath Explanation

IR		AR		D1		D2		PC			MUX 1	MUX 2	ALU	Memory	
In	out	In	out	In	out	In	out	In	out	Inc	Select	select	Code	Read	write
													1 1 1 1		

AR ← PC

0	0		1	0		0	0		0	0		0	1	0		1		1	1	0	0	0	0

IR ← M[AR], PC ← PC + 1

1	0		0	1		0	0		0	0		0	0	1		x		0	x	x	x	1	0

Control Word

Control Unit Organization

- Hardwired Control Unit
- Microprogrammed Control Unit

Hardwired Control Unit

Control logic is implemented with Gates, flip-flops, decoders and other digital circuits.

Advantage: Can be optimized to produce a faster mode of operation.

Disadvantage: Rearranging the wires among various components is difficult.

Updation in control logic is difficult.

Micro-Programmed Control Unit

Control logic is implemented with micro-programs.

Advantage: Updating the control logic is easy.
Disadvantage: Slower than hardwired control unit.

- All possible control words are stored in a memory **(Control Memory)** and based on the requirements, the specific control word is fetched from memory.

if assume,

CPU has = 16 instⁿs I1 to I16

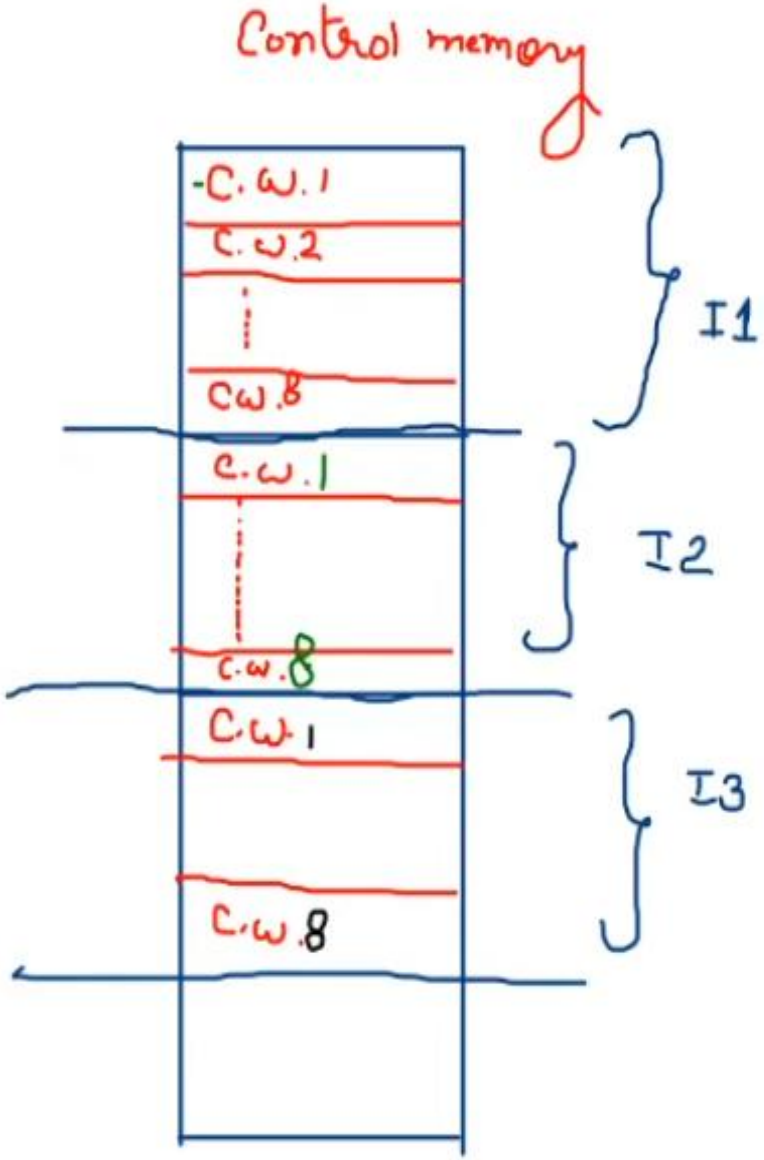
---

for each instⁿ
= 8 micro-oprs

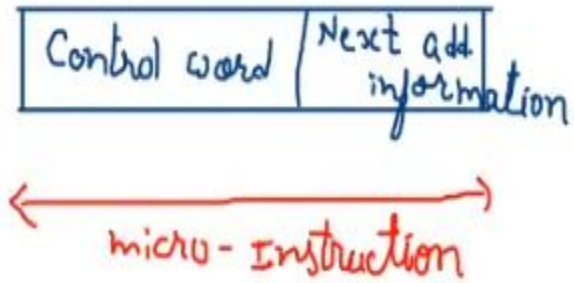
Total C.W's in memory = 16 * 8

$= 2^7 = 128$

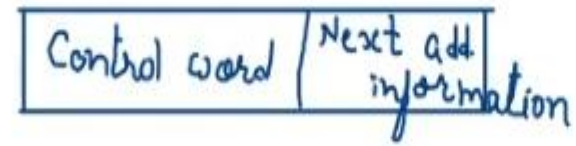
Control memory add. size = 7-bits

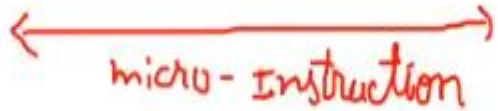


On each location in Control mem:-



On each location in Control mem:-





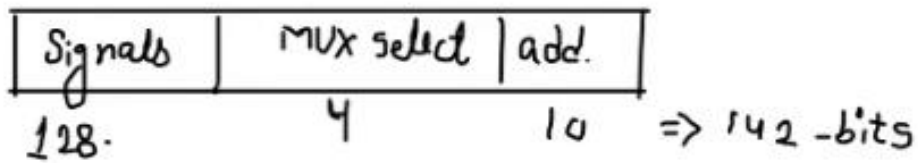
standard format of micro-instruction

Nano-programmed C.U.

Assume => Instructions = 16.

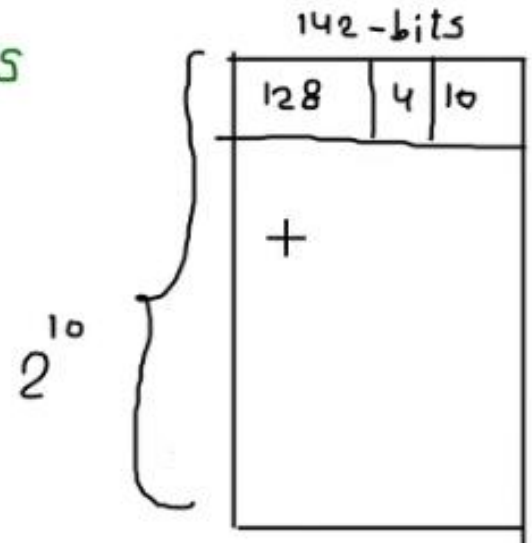
Each inst^{h} = 64 u-ops] → Total micro-operations = 16 * 64 = 2^{10} => add. = 10-bits

Each micro-inst^{h}



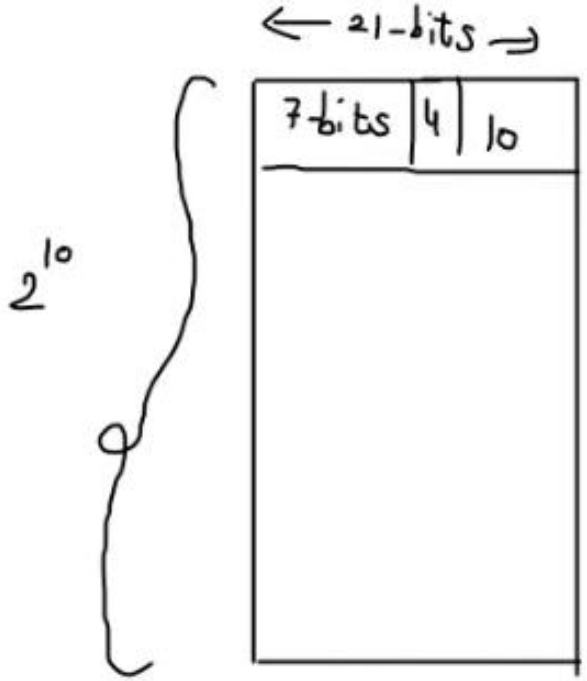
Control memory = 2^{10} * 142-bits

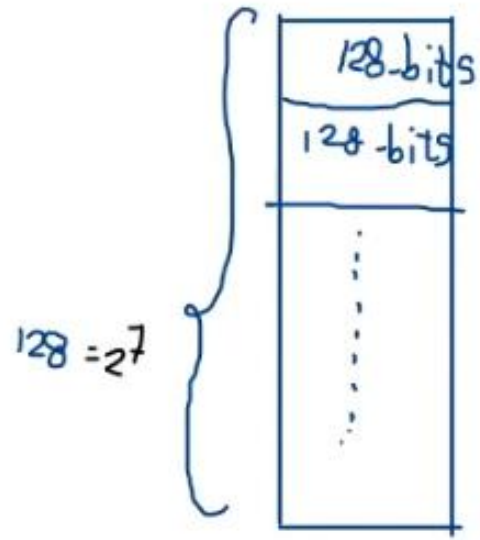
= 142 kbits



Assume, 128 - unique micro-operations

Nano-programmed C.U.





Unique Control words

$21 * 2^{10} = 21$ K bits

$2^7 * 2^7 = 2^{14} = 16$ K bits

Total size = 37 K bits

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.

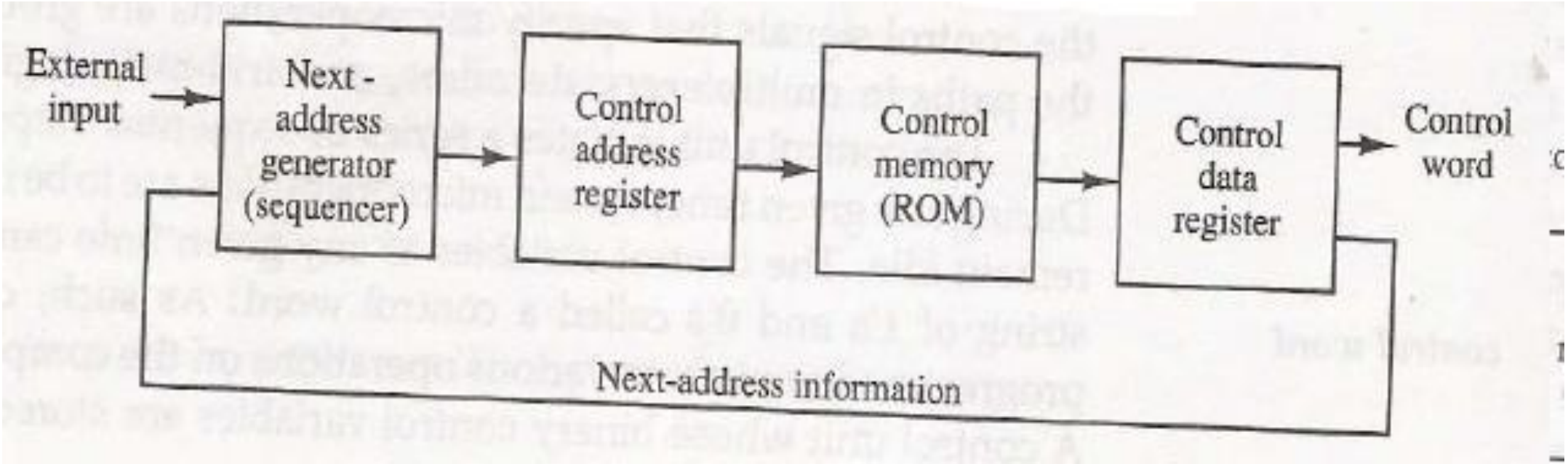


figure : Micro-programmed control organization

- 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**.

- 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.
- Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.

- The **next address generator is sometimes called a micro-program sequencer**, as it determines the address sequence that is read from control memory.

- Typical functions of a micro-program sequencer are incrementing the control address register by one, loading into the control address register an address from control memory, transferring an **external address**, or loading an initial address to start the control operations.

- 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**.

The steps of **Address Sequencing** in detail:

- 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.

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.

In summary, the address sequencing capabilities required in a control memory are:

1. **Incrementing** of the **control address register (CAR)**.
2. **Unconditional branch or conditional branch**, depending on status bit conditions.
3. A **mapping process** from the bits of the instruction to an address for control memory.
4. A facility for **subroutine call** and return.

Selection of address for control memory:

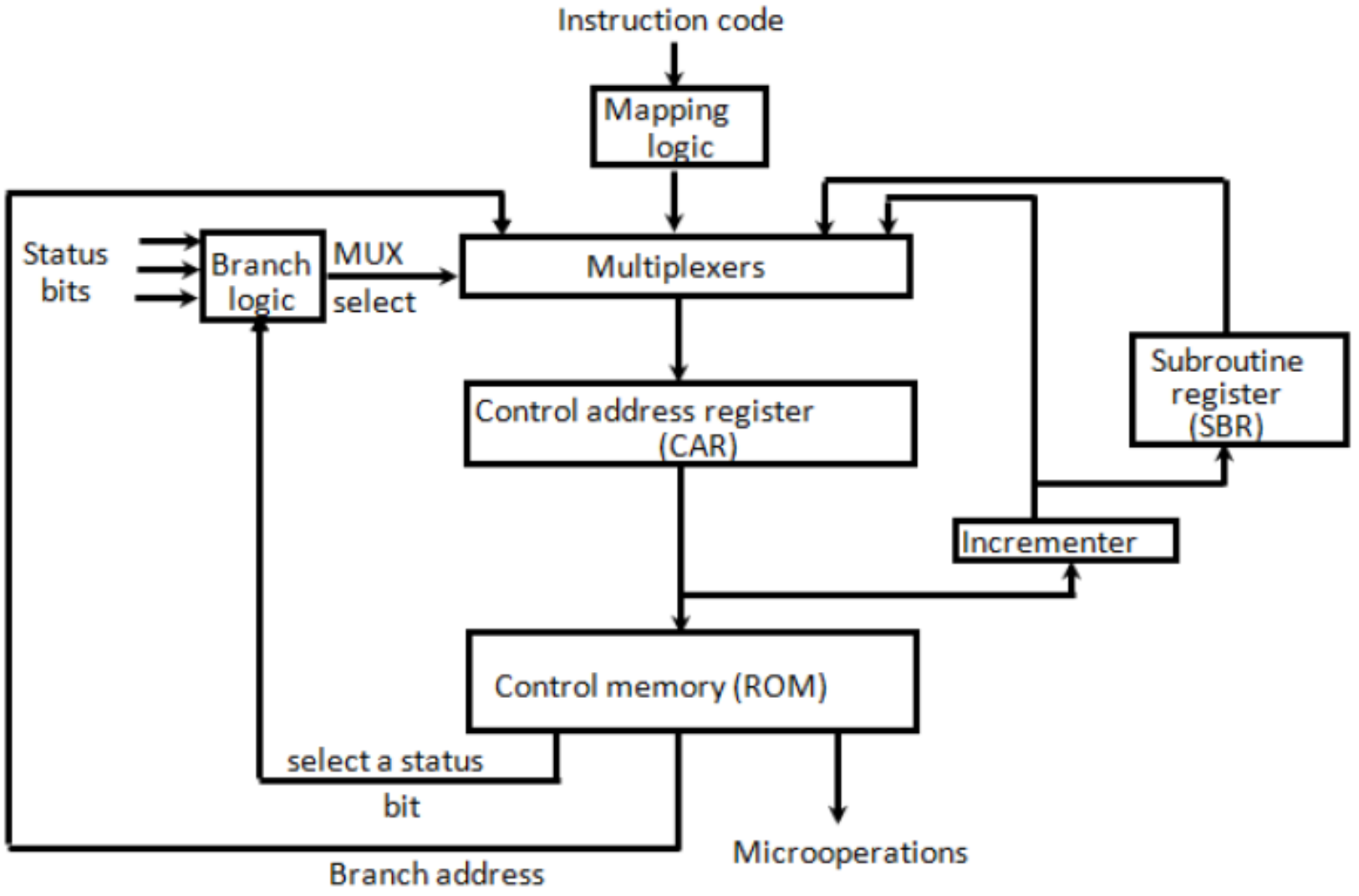


Figure 4.2: Selection of address for control memory

- Above figure 4.2 shows a block diagram of a control memory and the associated hardware needed for selecting the next microinstruction address.
- 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 **incrementor** 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 **branch logic** of figure 4.2 provides decision-making capabilities in the control unit.
- 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 an Instruction:

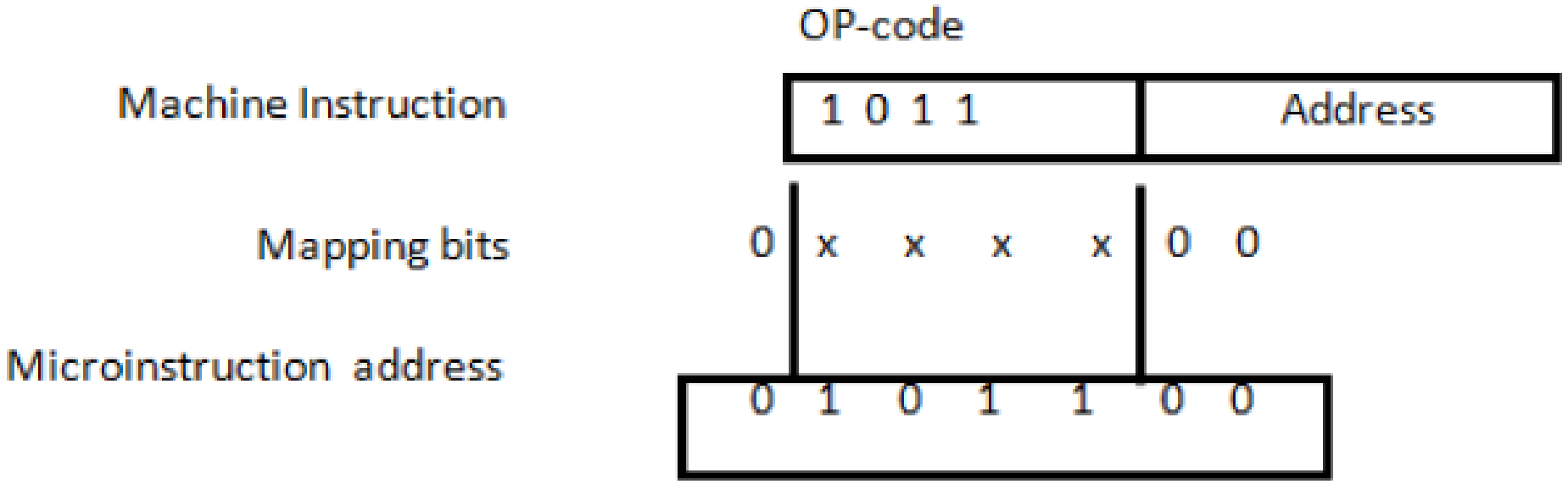


Figure 4.3: Mapping from instruction code to microinstruction address

- 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.
- 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.

- 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.

- 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.

Micro program control unit block diagram :

Computer Configuration

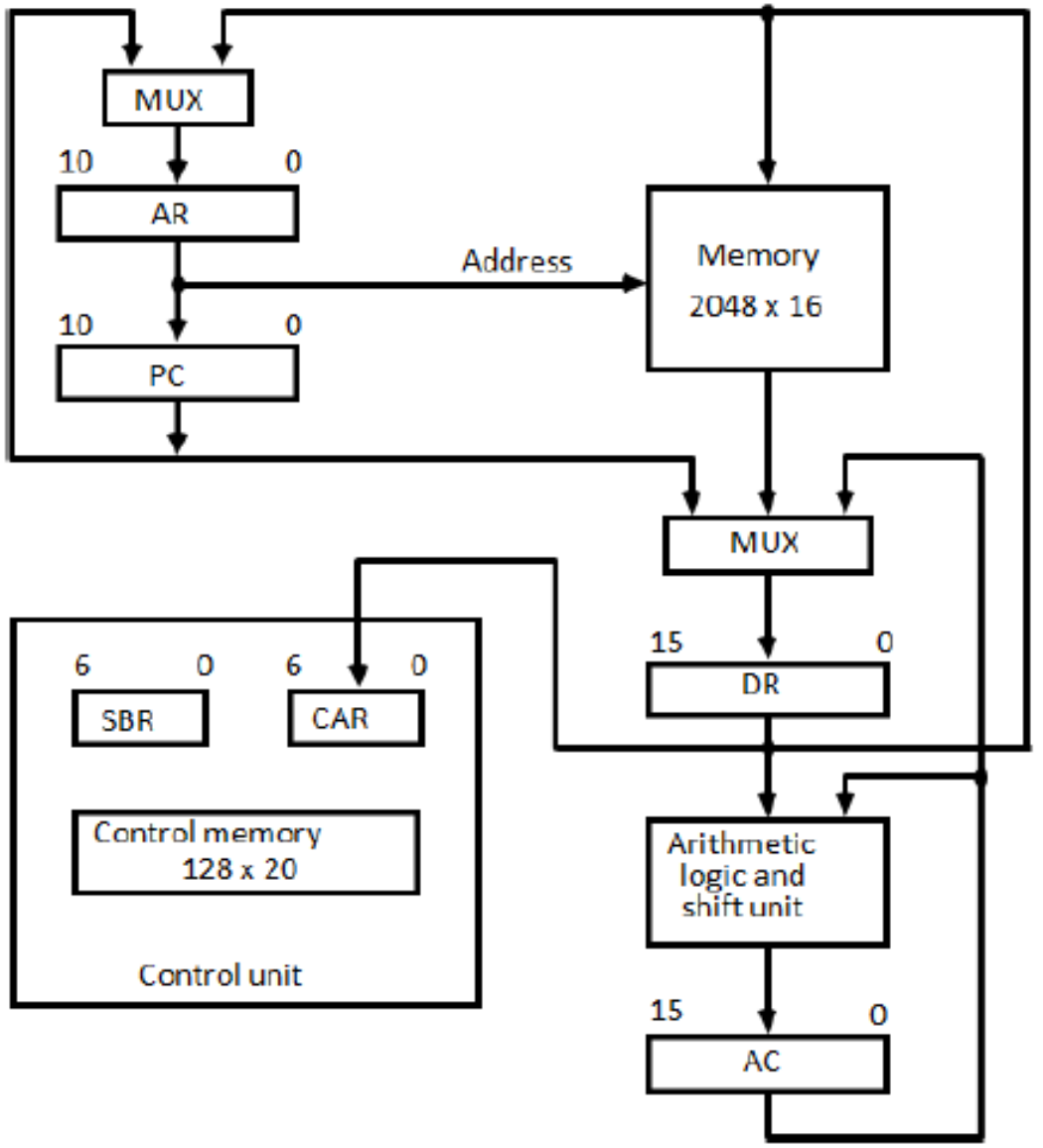


Figure 4.4: Computer hardware configuration

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.

Microinstruction Format :

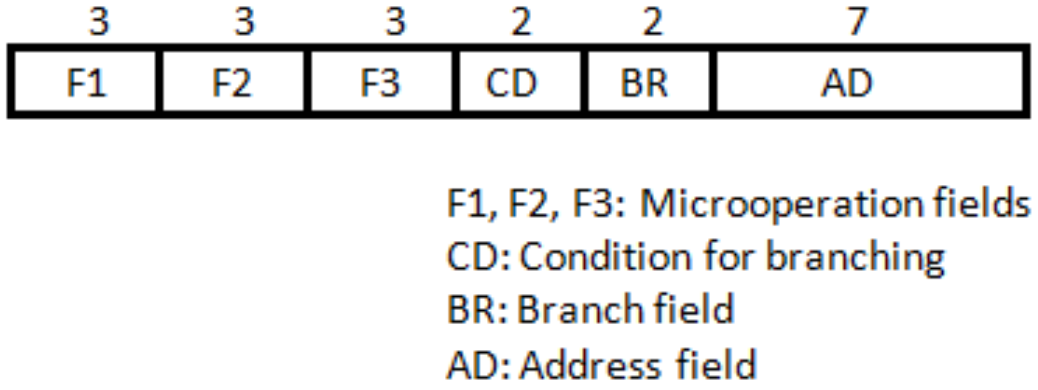


Figure 4.5: Microinstruction Format

The **20 bits** of the microinstruction are divided into four functional parts as follows:

1. The three fields F1, F2, and F3 specify microoperations for the computer.

The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.

2. The CD field selects status bit conditions.

3. The BR field specifies the type of branch to be used.

4. The AD field contains a branch address. The address field is seven bits wide, since the control memory has $128 = 2^7$ words.

- As an example, a microinstruction can specify two simultaneous microoperations from F2 and F3 and none from F1.

  DR ←M[AR] with F2 = 100

  PC ← PC + 1 with F3 = 101

- The nine bits of the microoperation fields will then be 000 100 101.
- The **CD (condition)** field consists of two bits which are encoded to specify four status bit conditions as listed in Table 4.1.

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

Table 4.1: Condition Field

• The **BR (branch)** field consists of two bits. It is used, in conjunction with the address field AD, to choose the address of the next microinstruction shown in Table 4.2.

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 4.2: Branch Field

Symbolic Microinstruction

- 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.

The fields specify the following Table

1.	Label	The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:).
2.	Microoperations	It consists of one, two, or three symbols, separated by commas, from those defined in Table 5.3. There may be no more than one symbol from each F field. The NOP symbol is used when the microinstruction has no microoperations. This will be translated by the assembler to nine zeros.
3.	CD	The CD field has one of the letters U, I, S, or Z.
4.	BR	The BR field contains one of the four symbols defined in Table 5.2.
5.	AD	The AD field specifies a value for the address field of the microinstruction in one of three possible ways: i. With a symbolic address, this must also appear as a label. ii. With the symbol NEXT to designate the next address in sequence. iii. When the BR field contains a RET or MAP symbol, the AD field is left empty and is converted to seven zeros by the assembler.

Micro programmed sequencer for a control memory:

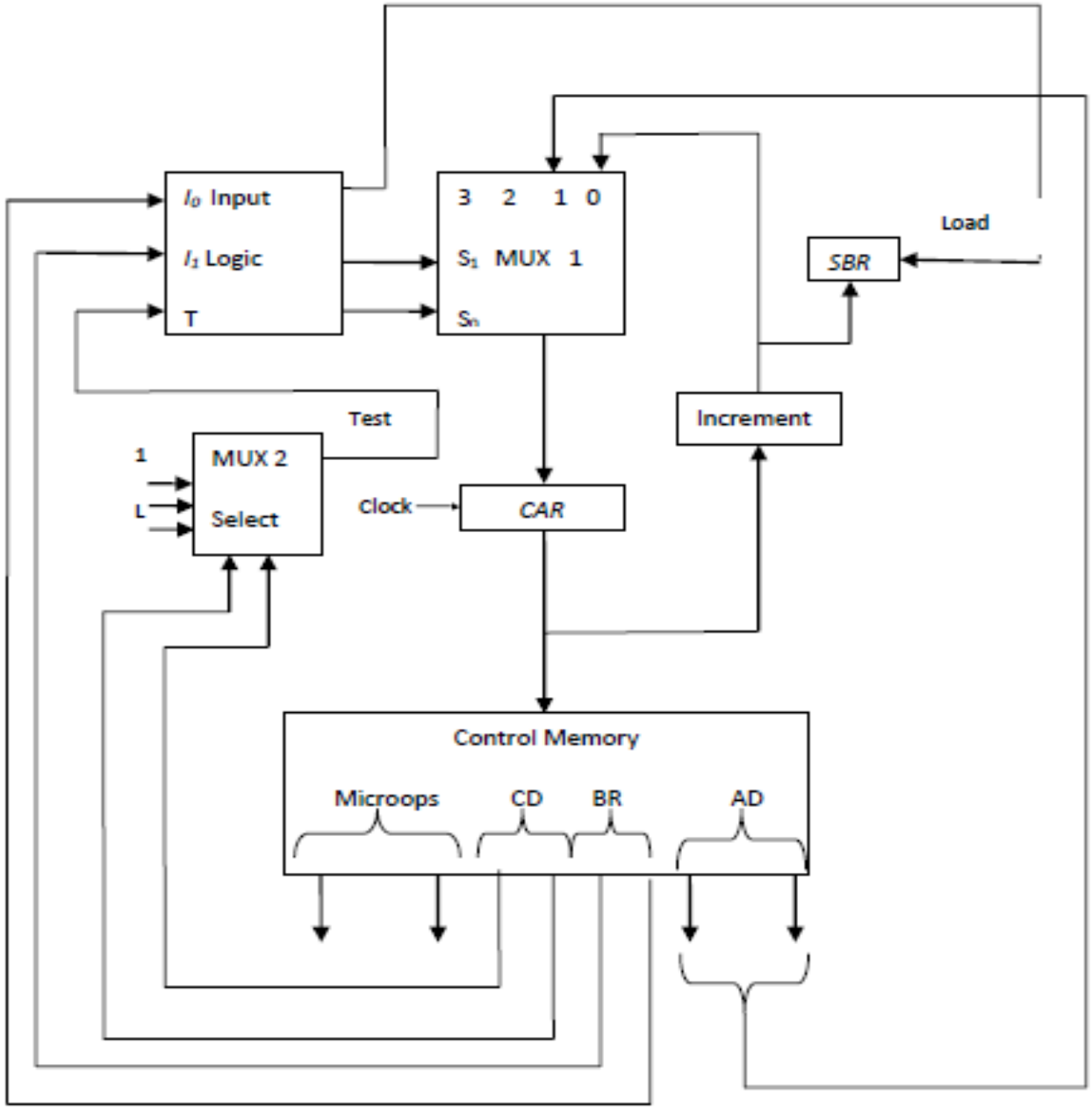


Figure 4.6: Microprogram Sequencer for a control memory

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

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

Input Logic : Truth Table

BR	Input			MUX 1		Load SBR
	I_1	I_0	T	S_1	S_0	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

Table 4.4: Input Logic Truth Table for 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.

- 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.

- The block diagram of the microprogram sequencer is shown in figure.
- 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.

- Although the figure shows 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.