Unit-4
Microprogrammed Control

⊛. Introduction:- (less imp)

Hardwired Control Unit → When the control signals are generated by hardware using conventional logic design techniques, then the control unit is said to be hardwired. It generates a specific sequence of control signals.

Microprogrammed Control Unit → A control unit whose binary control variables are stored in memory is called a micro-programmed control unit.

Control Memory → Control Memory is the storage in the microprogrammed control unit to store the microprogram.

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.

Microprogram → A sequence of micro instructions 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 content of the word in ROM at a given address specifies a microinstruction.

⊛. What is micro-instruction?

⇒ A computer instruction that activates the circuits necessary to perform a single machine operation is called micro-instruction.

Control address register → Control address register is a high speed circuit in a computing device that hold the addresses of data to be processed or of the next instruction to be executed. It specifies the address of the micro-instruction.

Sequencer → It is a part of the control unit of CPU. It generates the addresses used to step through the microprogram of a control store.

(less imp)

⊛ General Organization of micro programmed control unit:-

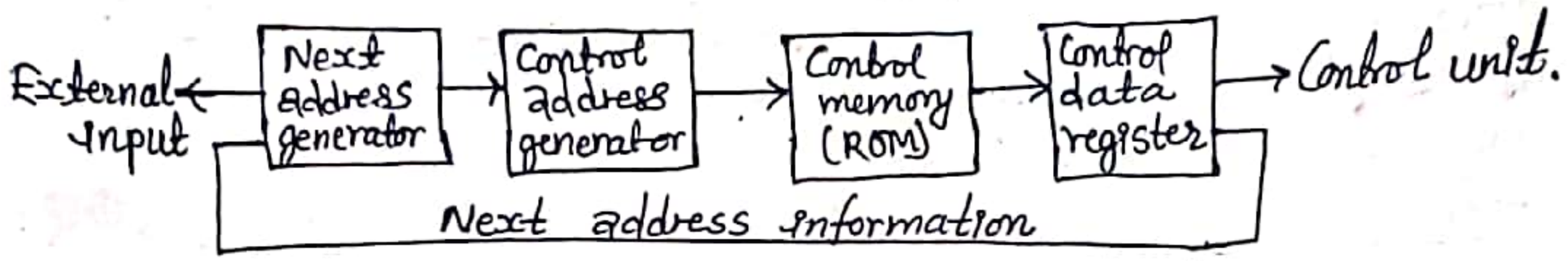


fig. Micro-programmed control organization.

→ The control memory is assumed to be a ROM, within which all the control information is permanently stored.
→ The next address generator is sometimes called a micro-program sequencer, as it determines address sequence that is read from control memory.
→ The data register is sometimes called a pip line register.

⊛ Address Sequencing:-

The address sequencing capabilities required in a control memory are:-

i) Incrementing of the control address register.
ii) Unconditional branch or conditional branch, depending on status bit conditions.
iii) A mapping process from the bits of the instruction to an address for control memory.
iv) A facility for subroutine call and return.

Selection of address for control memory:

The control address register (CAR) receives the address, from four different paths i.e, Control Memory (ROM), branch logic, multiplexer and subroutine register (SBR). The address is multiplexed with multiplexer before reaching to control address register (CAR).

The incrementer increments the content of the control address register by one, to select next micro instruction in sequence.

Conditional branching is obtained by using part of micro-instruction to select a specific status bit in order to determine its condition. The branch logic provides decision-making capabilities in the control unit. The following block diagram shows control memory and the associated hardware needed for selecting the next microinstruction address.

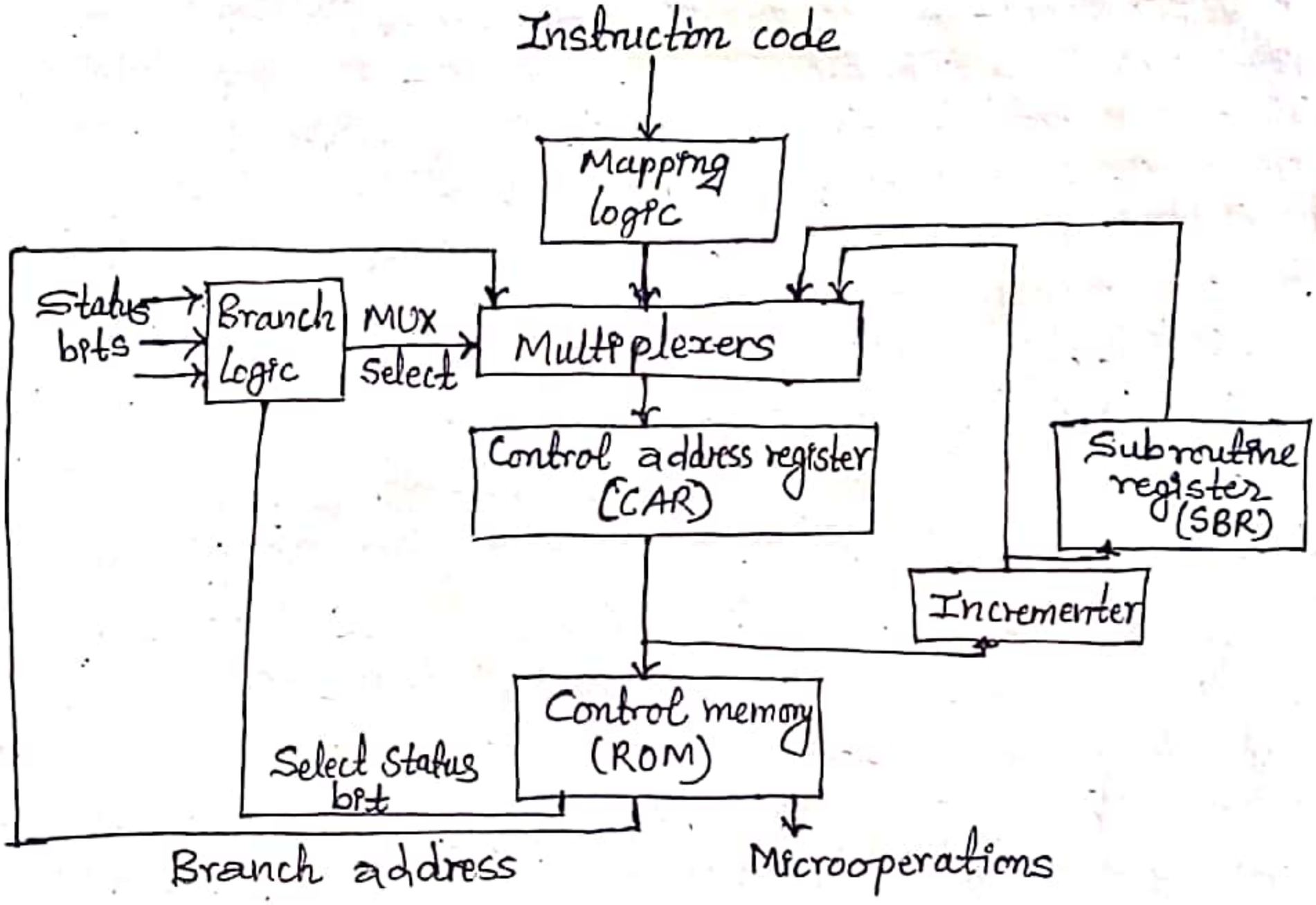


fig. Selection of address for control memory

⊛ Mapping of an instruction:- [Imp ; In model paper]

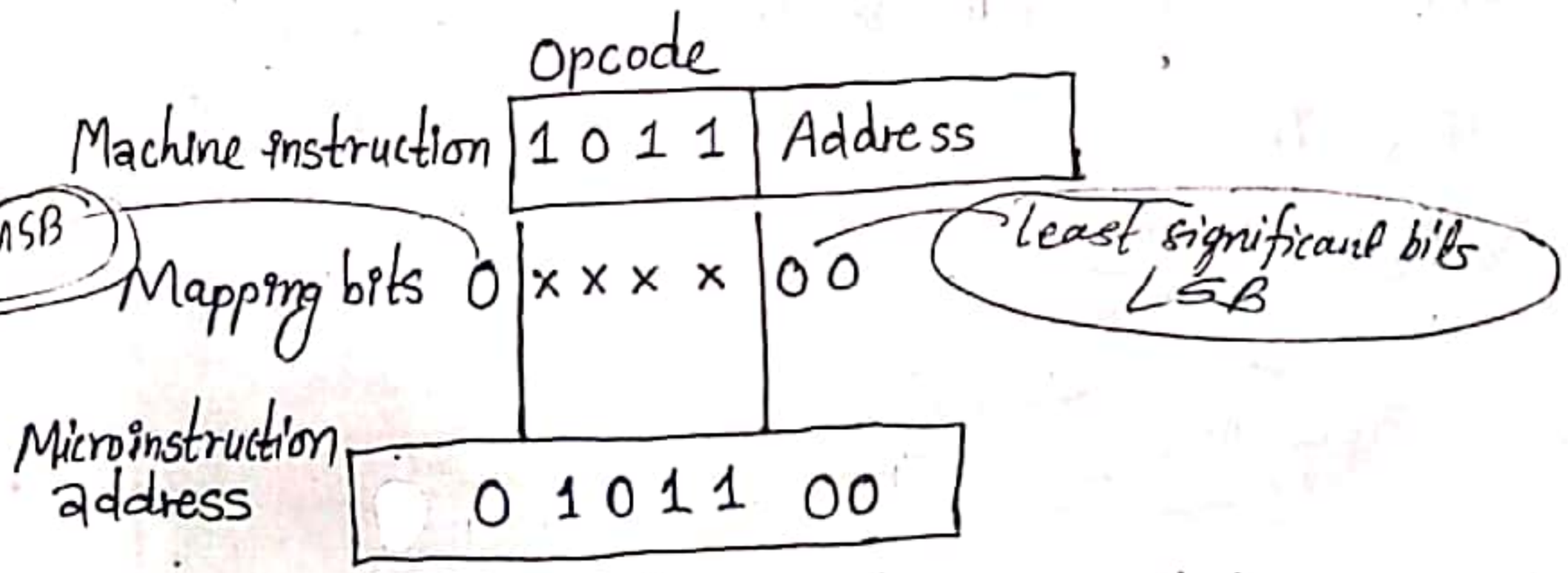


fig: Mapping from instruction code to microinstruction address.

The above figure illustrates a simple instruction format which has an operation code (opcode) of 4-bits which can specify upto 16 distinct instructions.

Assume further that the control memory has 128 words, requiring an address of seven bits. The above figure shows one simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory.

The mapping consists of placing a 0 in the most significant bit of the address, transfering the four operation code bits and clearing the two least significant bits 00 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.

⊛ Subroutines:- Subroutines are the programs that are used by other routines to accomplish particular task. Subroutines are written and stored seperately so whenever we require those subroutine we will ~~call~~ just call that subroutine and return to the main program. A subroutine can be called from any point within the main body of the microprogram. Whenever subroutine is called it halts the main program and it provide returning to the ~~main program~~ same point. ~~in main program~~. It provides control to subroutine and execute the subroutine and finally returns to main program.

→ similar to functions

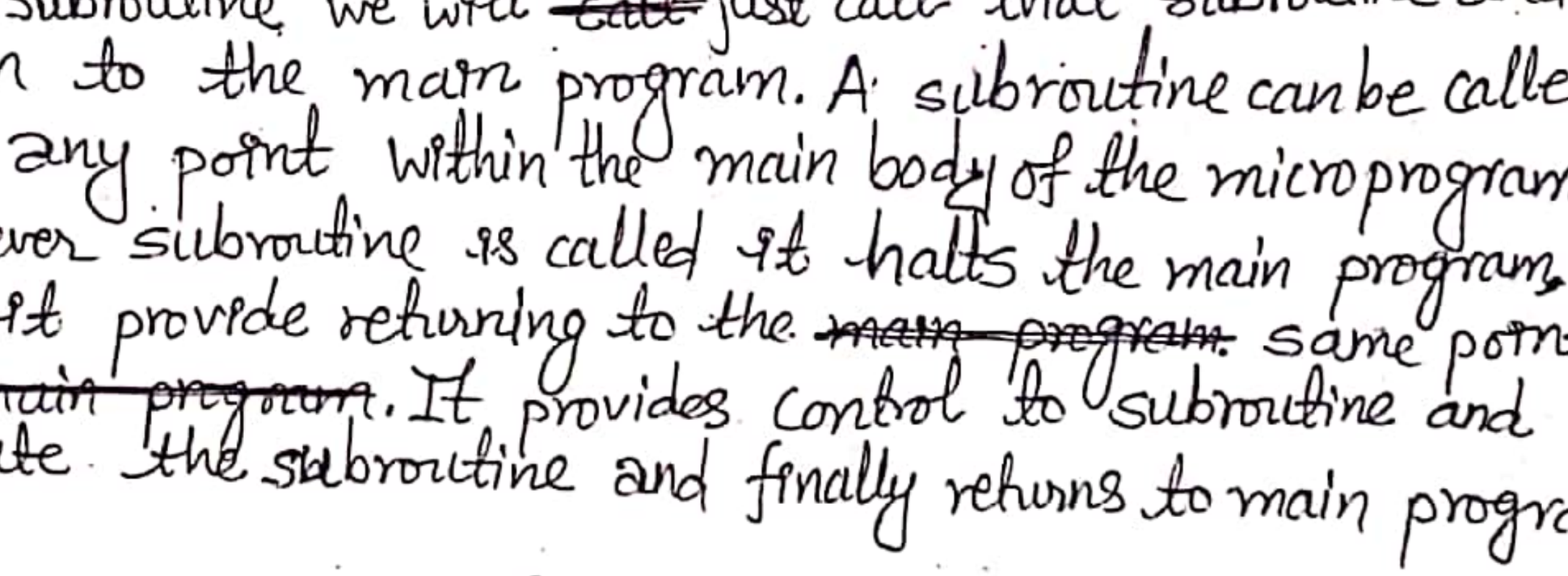
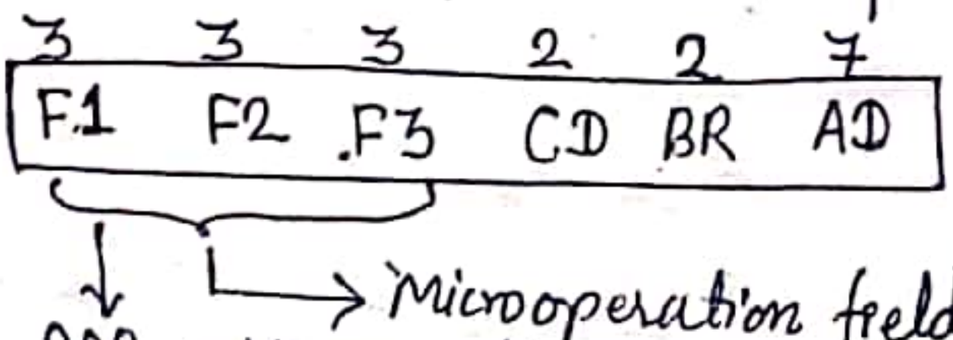


fig. Process for calling of subroutine by main program

⊛. Microinstruction Format:-

→ Microinstruction format consist of 20-bits in which F_1, F_2 and F_3 are microoperation fields each of 3-bits. If any particular microoperation field is 000 then it specifies no operation.



CD: Condition for branching (2-bits)
BR: Branch field (2-bits)
AD: Address field. (7-bits)

fig. Microinstruction Format

→ 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 = 27 words.

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: Condition field

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 ← SR (Return from subroutine)
11	MAP	CAR ← DR (11-14), CAR (0, 1, 6) ← 0

Table: 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:-

S.N.	Fields	Comments
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 seperated by commas. 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 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 JMP, CALL, RET or MAP.
5.	AD	The AD field specifies a value for the address field of microinstruction in one of following three possible ways; → With a symbol NEXT to designate the next address in sequence. → With a symbolic address, this must also appear as label. → 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.

Table: Symbolic Microinstruction

⊛ Design of Control Unit:

⊛. Field Decoding:

Since there are three microoperation fields (F1, F2 and F3) so we need 3 decorders. Only some of the outputs of decorders are shown to be connected to their output. Each of the output of the decorders must be connected to the proper circuit to initiate the corresponding microoperation.

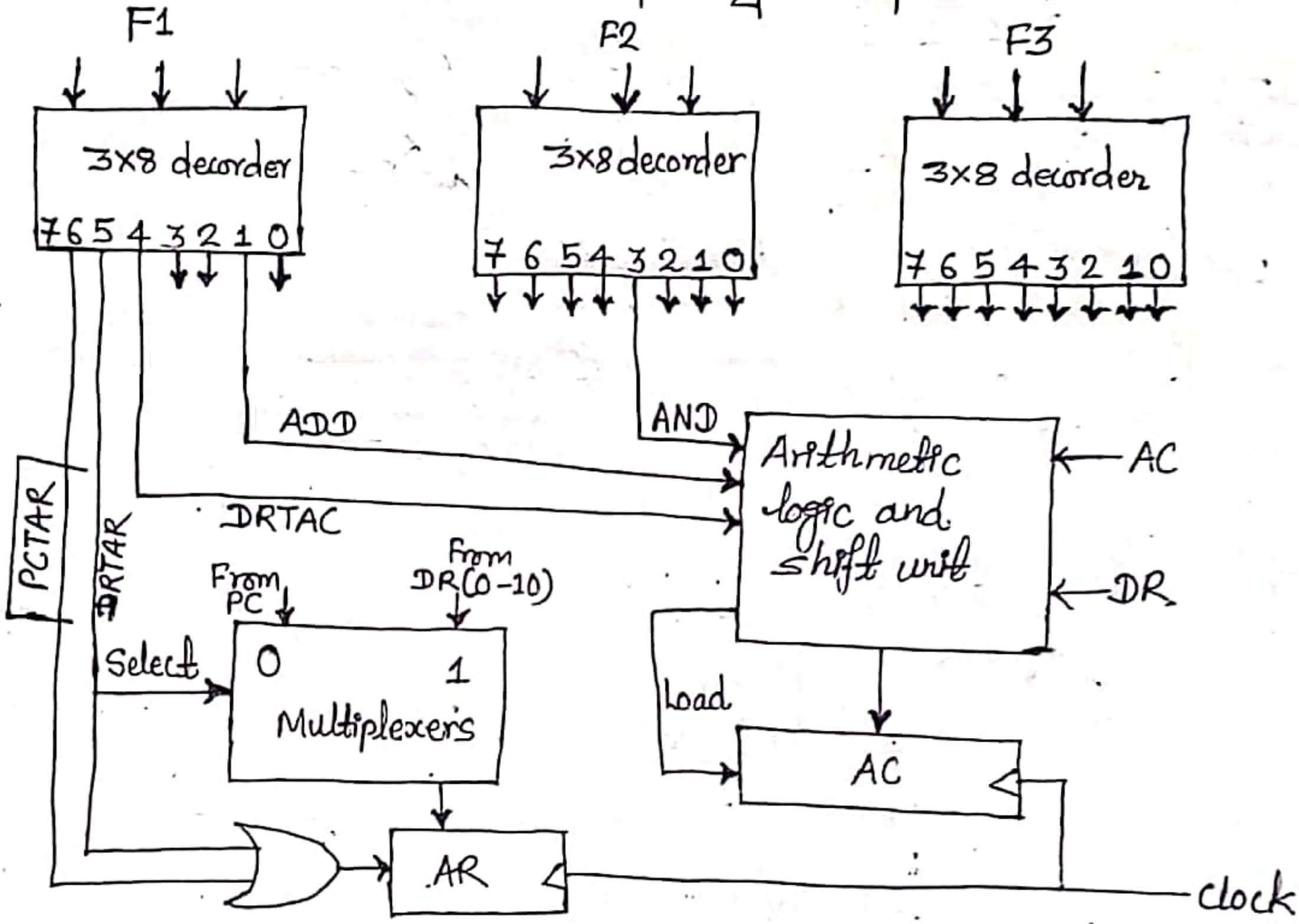


Figure: Decoding of Microoperation fields

⊛. Micro programmed sequencer for a control memory: [Imp]

Micro programmed sequencer is a part of micro programmed control unit that generates next address of instruction which is going to be executed. The basic components of micro programmed control unit are the control memory and the circuits that select the next address.

The next address logic of the sequencer determines the specific address source to be loaded into the

control address register, (CAR).

Two important factors that must be considered while designing the microprogrammed sequencer are;

i) The size of the microinstruction.

ii) The address generation time.

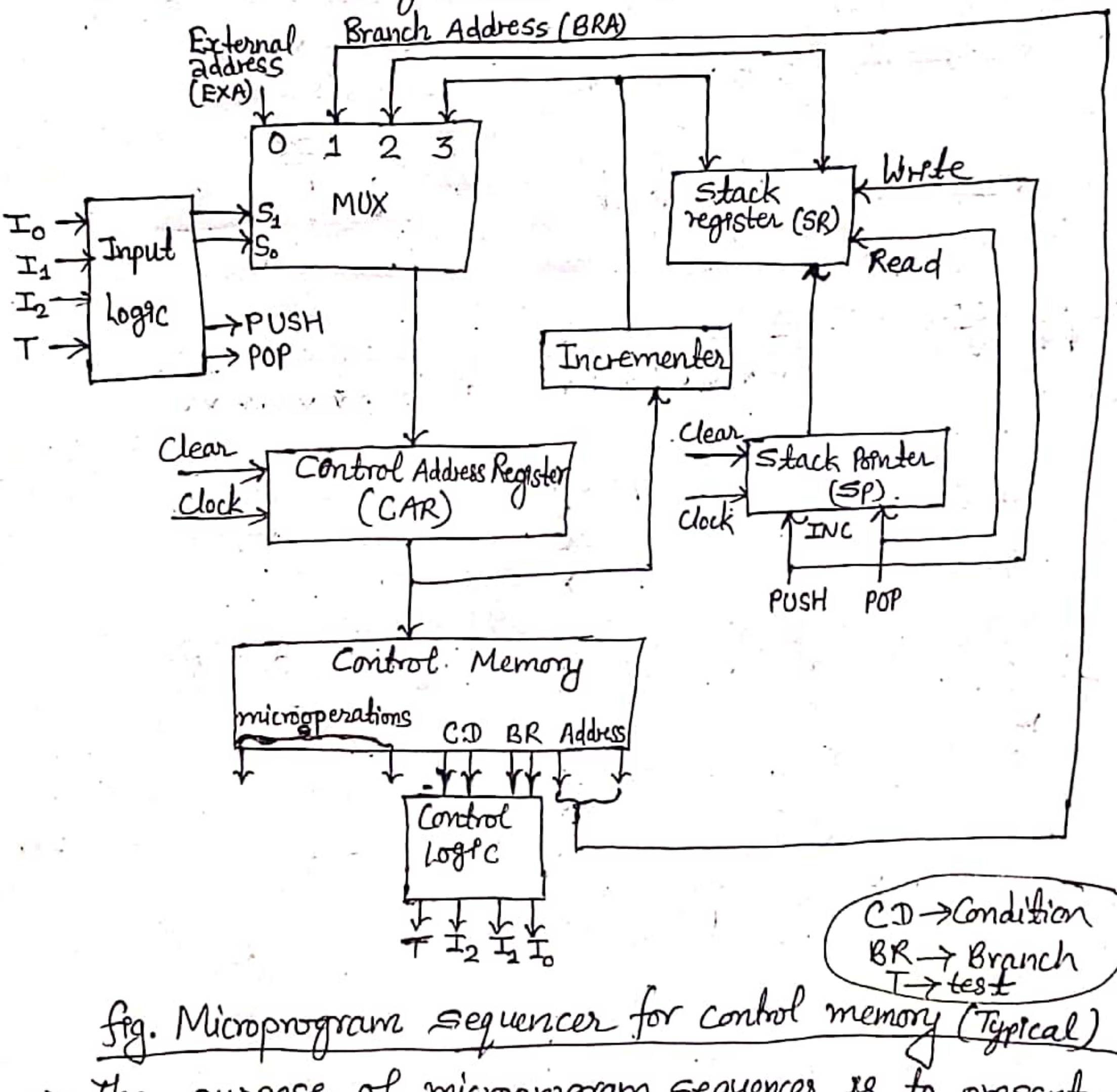


fig. Microprogram sequencer for control memory (Typical)

→ The purpose of microprogram sequencer is to present an address to the control memory so that a microinstruction may be read and executed.

→ Here multiplexer selects an address from 4 sources and routes it into a control address register

→ The output from CAR provides the address for control memory.

→ The contents of CAR are incremented and applied to the multiplexer & to the stack register file.
→ The register selected in the stack is determined by stack pointer.

BR	Input			MUX		Load SBR
	I_1	I_0	T	S_1	S_0	L
00	0	0	0	0	0	0
00	0	0	1	0	1	0
01	0	1	0	0	0	0
01	0	1	1	0	1	1
10	1	0	x	1	0	0
11	1	1	x	1	1	0

Table: Truth table for microprogram sequencer

⇒ Typical sequencer operations are: increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack and other address sequencing operations. With three inputs, the sequencer can provide up to eight address sequencing operations. Some commercial sequencers have three or four inputs in addition to the T input and thus provide a wider range of operations.

⊛. Differences between Handwired control unit and microprogrammed control unit.

Handwired Control Unit	Microprogrammed Control Unit
i) Handwired control unit generates the control signals needed for the processor using logic circuits.	i) Microprogrammed control unit generates the control signals with the help of micro instructions stored in control memory.
ii) It is faster compared to microprogrammed control unit as the required control signals are generated with the help of hardwares.	ii) This is slower as micro instructions are used for generating signals.
iii) It is difficult to modify as the control signals that we need to be generated are chard wired.	iii) Easy to modify as the modification need to be done only at the instruction level.
iv) It is expensive as everything has to be realized in terms of logic gates.	iv) It is less expensive as only micro instructions are used for generating control signals.
v) It can not handle complex instructions	v) It can handle complex instructions
vi) Used in Reduced Instruction Set Computers (RISC).	vi) Used in Complex Instruction Set Computers (CISC).