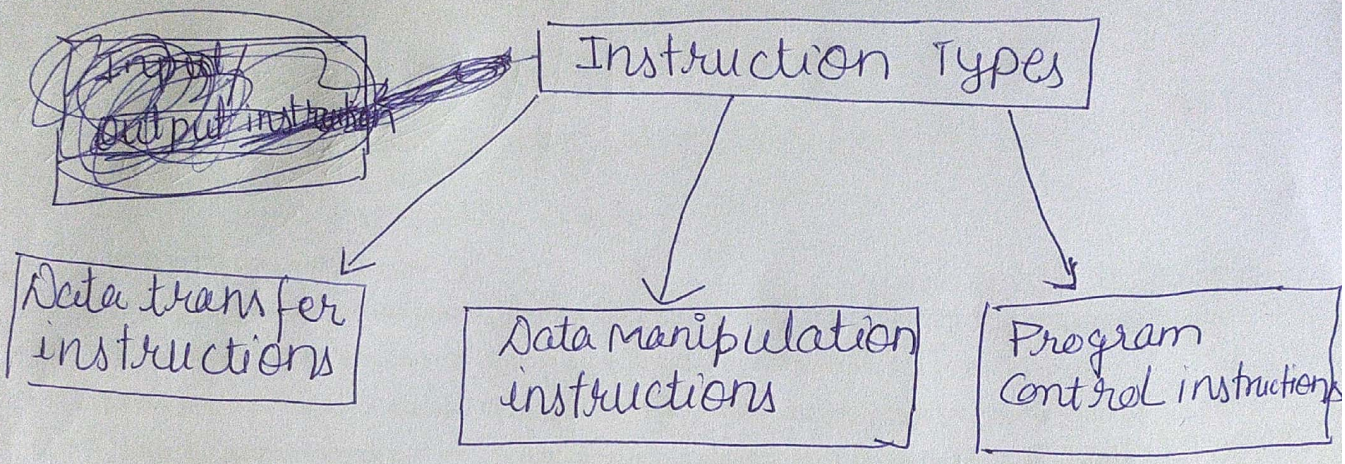
Unit-3 Control Unit

Types of instruction

There are three types of instruction



① Data transfer instruction ⇒ Data transfer instructions are those instructions that transfers the data from one location to another (Register, Memory) without changing the data content.

⇒ These transfers can be between the two processor register or between the memory location or register to memory or register to input or output devices.

⇒ example :

Name	Mnemonic
Load	LD
Store	ST
MOVE	MOV
EXCHANGE	XCH
INPUT	IN
OUTPUT	OUT
PUSH	PUSH
POP	POP

⇒ Load instruction : used to transfer a data from memory location to processor register called an Accumulator register (AC)

Store: used to transfer data from Accumulator register to some memory location.

MOVE: used to transfer the data content from one processor register to another or memory.

Exchange: used to swap information between two register or memory and register.

Input and output: used to transfer data from register and input or output devices.

PUSH and POP : used to transfer data content between processor register and memory stack

② Data Manipulation Instruction ⇒

Data manipulation instructions are those instructions that perform arithmetic, logic or shift operations to manipulate data. So, these are three types

ⓐ Arithmetic instructions
ⓑ logic instructions
ⓒ Shift instructions.

ⓐ Arithmetic instructions : It performs arithmetic operations on data.

ex ⇒

Name of instruction	Mnemonic
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Increment	INC
Decrement	DEC

Logic instructions ⇒ These instructions are used to perform the logical operations on data

ex:

Name of instruction	Mnemonic
AND	AND
OR	OR
Complement	COM
Exclusive-OR	XOR
Exclusive-NOR	XNOR
clear	CLR

Shift instructions ⇒ These instructions are used to shift data of register.

ex⇒

Name of instruction	Mnemonic
Logical shift left	SHL
Logical shift right	SHR
Rotate left	ROL
Rotate right	ROR
Arithmetic shift left	SHLA
Arithmetic shift right	SHRA

Program Control Instructions :⇒ Program control instructions are those instructions that can alter the flow of control.

ex⇒

Name of instruction	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare	CMP

~~Mathematical instructions ⇒ these instructions are for~~

Instruction formats :

refer to unit-1

Microoperation :- refer to unit-1

~~Instruction cycle~~

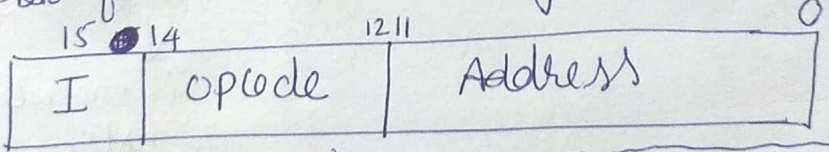
Basic computer instructions

⇒ The basic computer has 16-bit instruction Register (IR)

⇒ It has three instruction code formats which are:

1. Memory-reference instruction
2. Register-reference instruction
3. input-output instruction

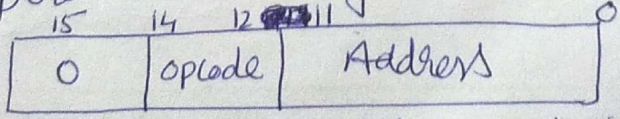
① Memory-reference instruction ⇒ In basic computer memory-reference instruction uses 12-bits to specify memory address, three-bit for opcode (operation code) and 1-bit for addressing mode for direct and Indirect Addresing mode.



I = 0 (direct addressing mode)
I = 1 (Indirect addressing mode)

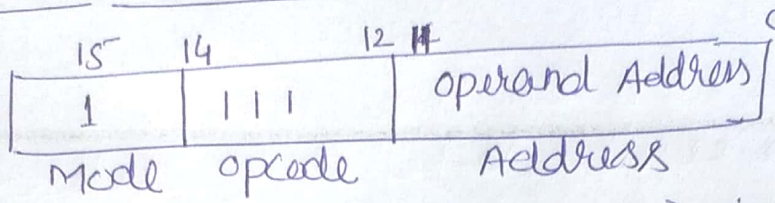
opcode is 3-bits which uses 000 to 110

② Register-reference instruction ⇒ These instructions perform operations on register rather than memory.



Here, opcode (12-14) is 111 (differentiate it ~~memo~~ from memory reference and IR(15) is 0 (differentiate it from I/O instructions)

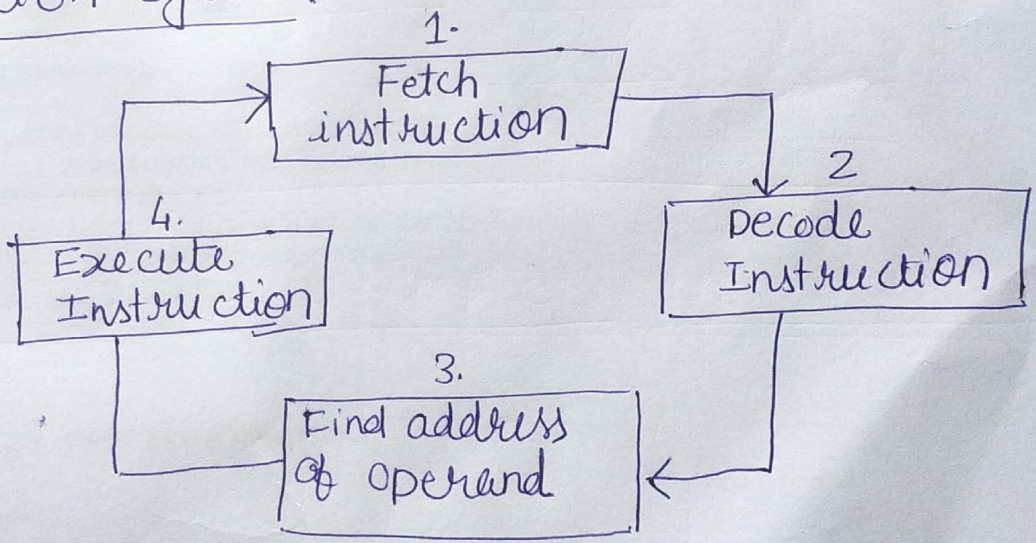
③ Input/output instruction :



In this instruction opcode (14–12) is 111 (differentiate it from memory-reference) and IR(15) is 1 (differentiate it from ~~reference~~ register reference instructions)

Instruction cycle

⇒ The time taken by the CPU to fetch and execute one single instruction is called instruction cycle.



Instruction cycle

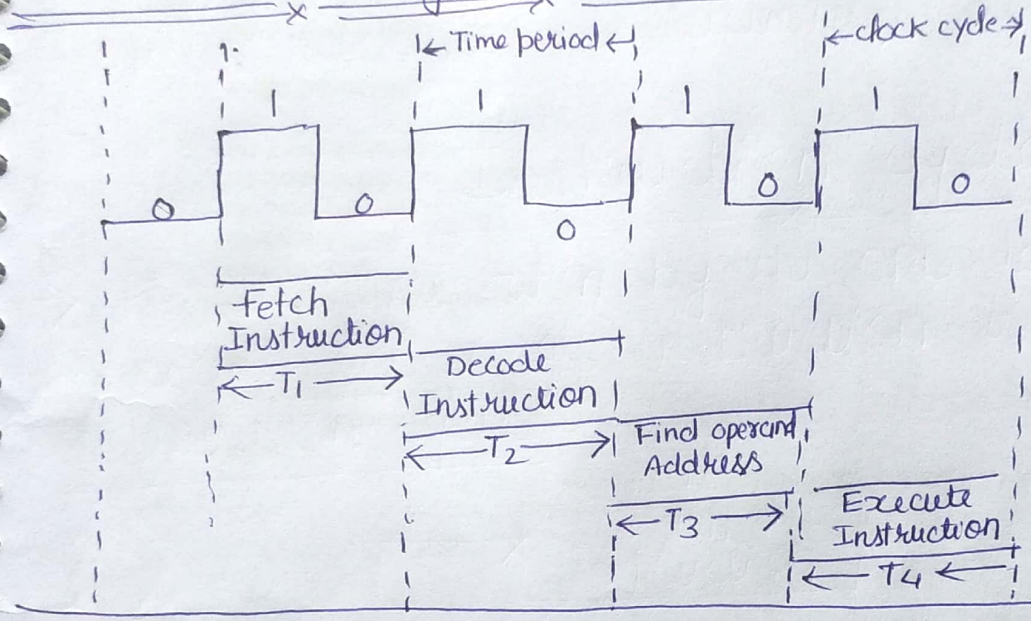
Instruction cycle steps

Step① - Fetch: The CPU fetches instruction from memory.
Step② - Decode: The control unit decode the instruction
Step③ - Find address of operand: using addressing modes find the operand effective address
Step④ - Execute: The ALU executes instruction and operate on data.

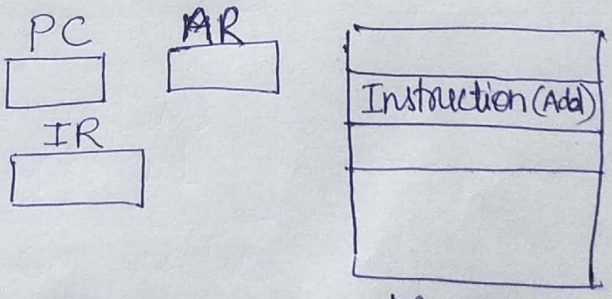
CPU clock cycle ⇒ The CPU speed depends upon the number of clock cycle needed to execute the instruction.

Clock cycle ⇒ A clock cycle is the time taken between the two pulses of an oscillator

Instruction cycle steps in clock cycle



Execution of a complete instruction



⇒ Initially, the program counter PC is loaded with the address of the first instruction in the program.

⇒ The sequence counter (SC) is cleared to 0. After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T0, T1, and so on.

T_0: AR ← PC // Transfer the address of instruc
// from PC to AR with timing signal T_0

T_1: IR ← M[AR], PC ← PC+1 // The instruction read
// from memory is then placed in the
// IR (Instruction register) and at the
// same time PC is incremented by one to
// prepare it for the address of the next
// instruction in the program.

T_2: D_0 ..... D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)
// At time T_2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.

⇒ During time T_3, the control unit determines the types of instruction. The three possible instruction types available in the basic computer.

⇒ Decoder output D_7 is equal to 1 if the operation code is equal to binary 111. As in flow chart next page if $D_7 = 1$, the instruction must be register-reference or input-output type.

⇒ If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.

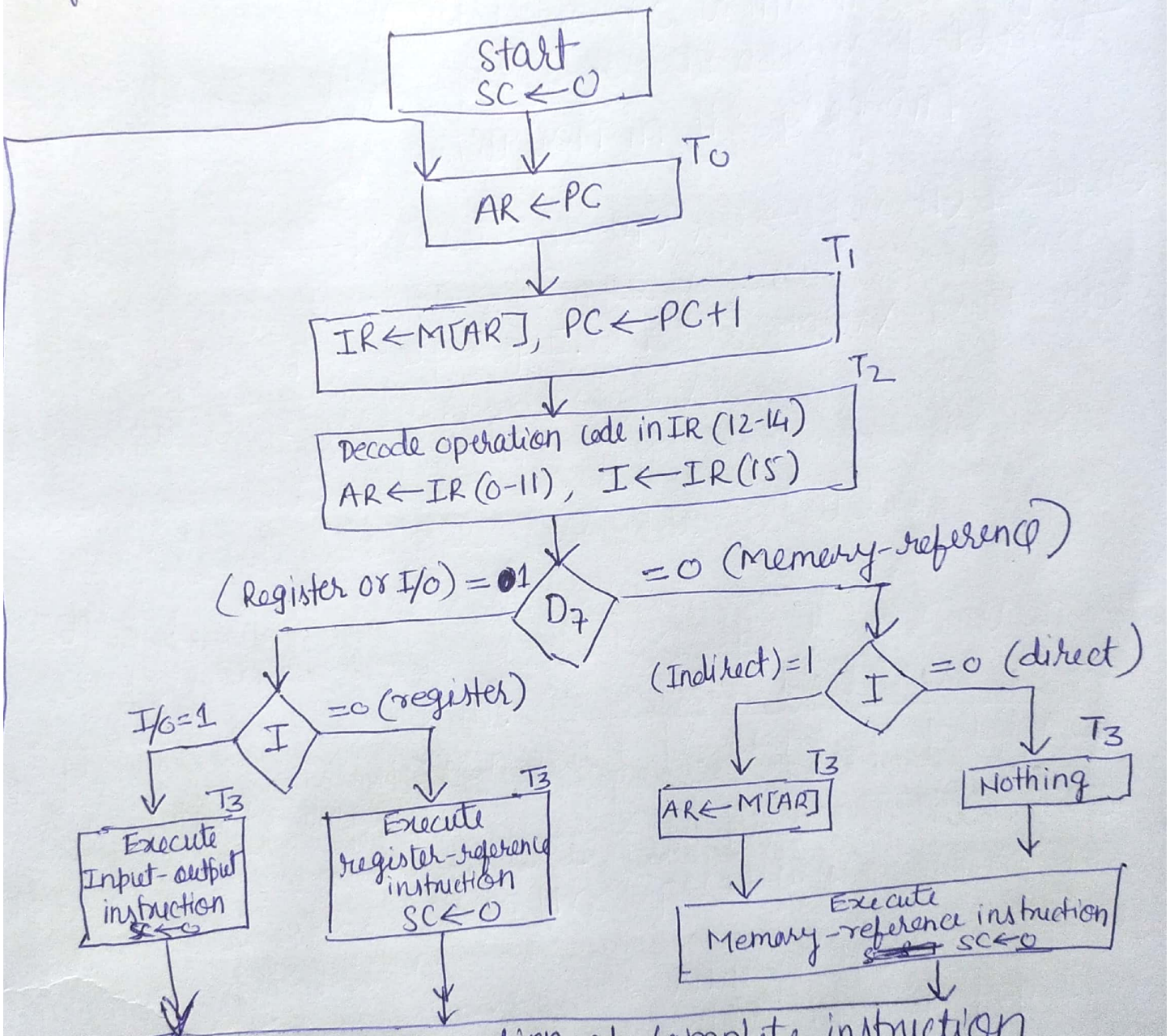
⇒ Control unit then inspects the value of the first bit of the instruction which is now available in flip-flop I.

⇒ If $D_7 = 0$ and I = 1, we have a memory-reference instruction with an indirect address. It is then necessary to read the effective address of operand from memory.

AR ← M[AR] // Indirect addressing.

⇒ When a memory-reference instruction with I = 0, then it is not necessary to do anything since the effective address is already in AR.

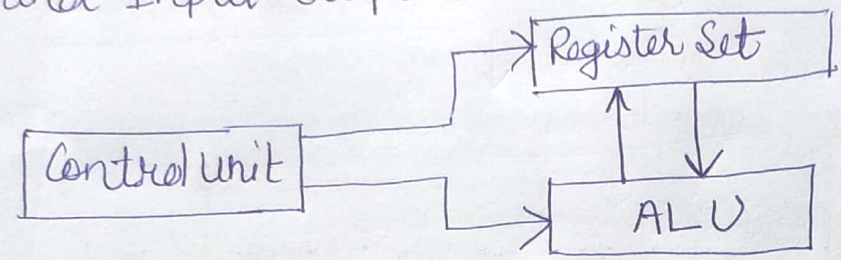
when $D_7 = 1$ and $I = 0$ then Register-reference instructions are found. The 12-bits available in IR(0-11) are also transferred to AR during time T_2.
These instructions are executed with the timing signal T_3.
When $D_7 = 1$ and $I = 1$, then input-output instructions are found. and executed with the timing signal T_3.



Flow chart for execution of complete instruction

Control unit

⇒ The control unit is the brain or nerve center of the computer hardware. It keeps track of the instruction cycle and generate relevant control signal at appropriate time so that appropriate microoperations are done in the CPU and other units such as memory and Input-output.

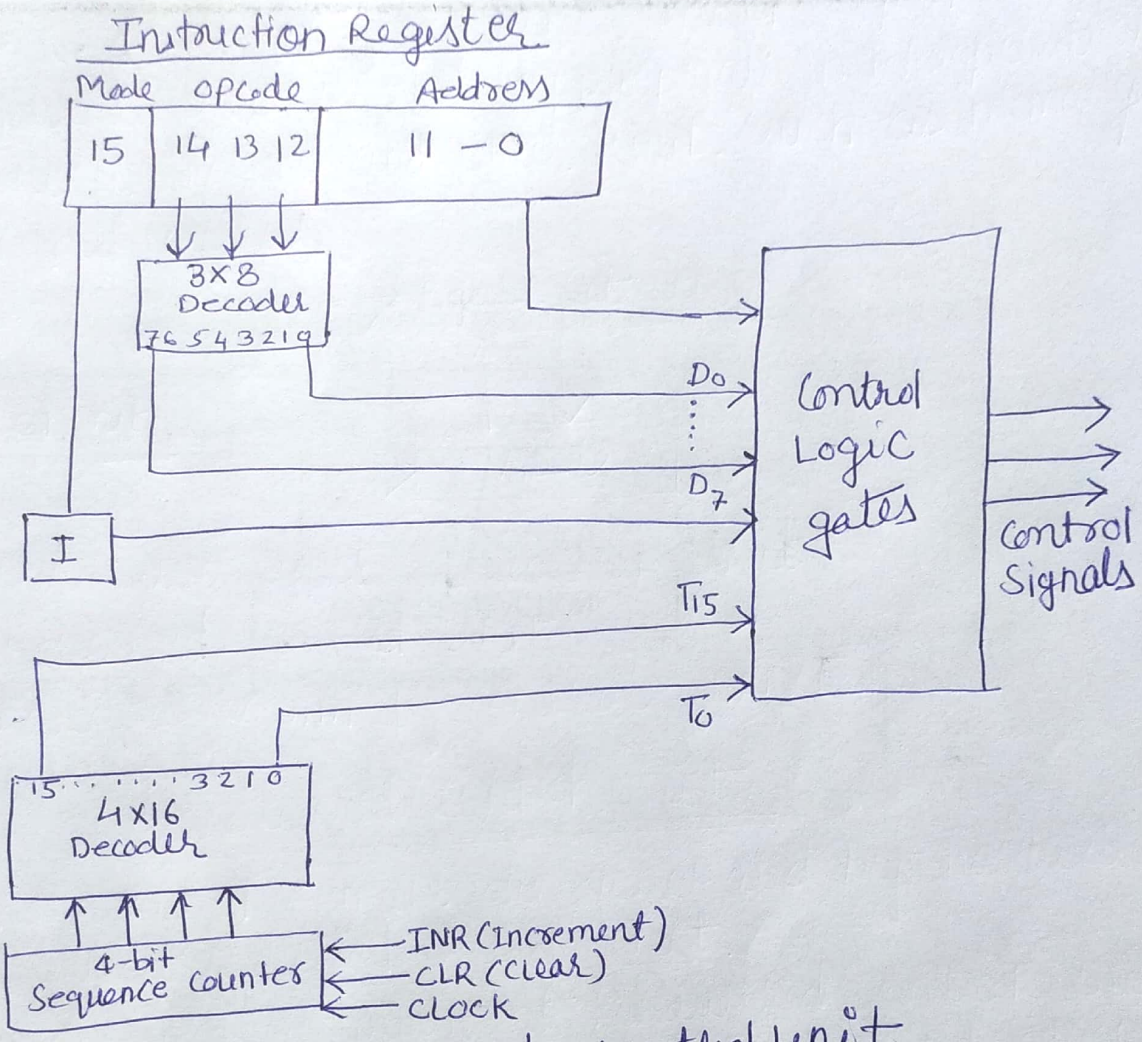


⇒ There are two types of control unit
(i) Hardwired control unit
(ii) Microprogrammed control unit

Hardwired control unit

⇒ When the control signals are generated by hardware it is called hardwired control unit.
⇒ Hardware used to implement hardwired control unit are Logic gates, flip-flops, decoders and other digital circuits.
⇒ It consist of two decoder, a sequence counter and a number of control logic gates.
⇒ Instruction read from memory is placed in the instruction register (IR) which is divided into three parts The I bit for addressing mode, the opration code bits (12-14) and operand addres bits (0-11).
⇒ Bit 15 of instruction is transferred to flip-flop (I).
⇒ The opcode bits (12-14) are decoded with 3×8 decoder
⇒

⇒ The 4-bit sequence counter can count in binary from 0 through 15, which are decoded in 16 timing signals T_0 to T_{15} as shown in fig below.



Advantages of Hardwired control unit

1. It generate signal with the help of hardware so Hardwired control unit is fast.
2. If instruction set is simple and number of instructions are less, then hardwired control unit is used.
   For example; RISC (Reduced instruction set computer) uses hardwired control unit.

Disadvantages of Hardwired control

① It uses hardware for generating control signals so, it's cost is very high.

② For complex instruction set. i.e. when there are large number of instruction set this control unit can't be used.

③ If any modification is required in hardwired control unit it is very difficult to make changes like if there is a design mistake or a new instruction need to add to an existing design or a new hardware component of higher speed is available.

~~Control variable~~ :- ~~String of 1's and 0's are called control variable.~~

~~Microcode or codeword~~ :-

Control Signal ⇒ It is a group bits to select the path.

Control variables :- A binary variable 0's and 1's specify a microoperation called control variables.

Control word or Codeword or microcode :- String of 0's and 1's which represents the control variables is called control word, or codeword or microcode.

Control Memory ⇒ The control signals needed for an ~~instruction~~ microinstruction can be stored in the memory known as control memory.

⇒ By fetching the control memory words one-by one, the control signals can be generated.

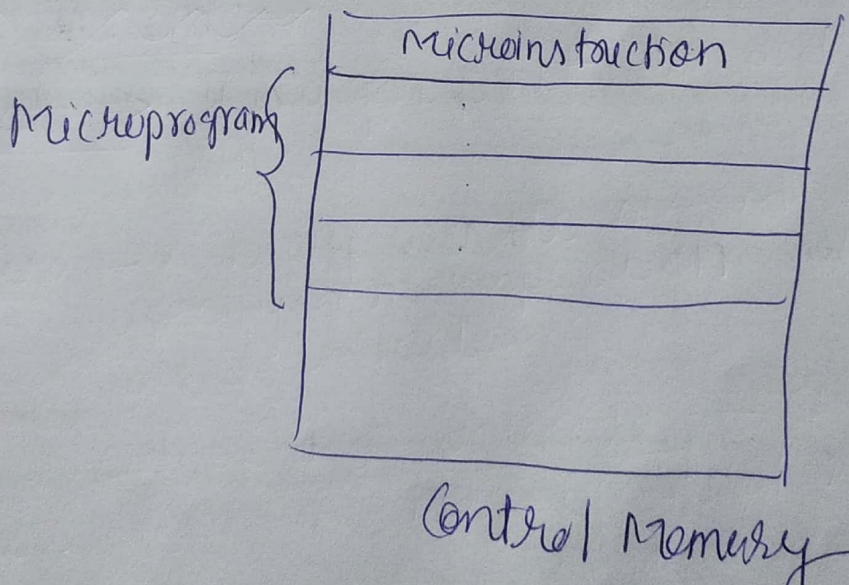
⇒ The control memory is a Read only memory (ROM).

Microinstruction ⇒ If we encode the signals for each step as control word or codeword or microcode, then each such codeword is called Microinstruction

⇒ Each control memory word is known as a microinstruction.

Microroutine :- The sequence of codewords for an microinstruction is called microroutine.

Microprogram :- In control memory set of microinstruction is called microprogram.



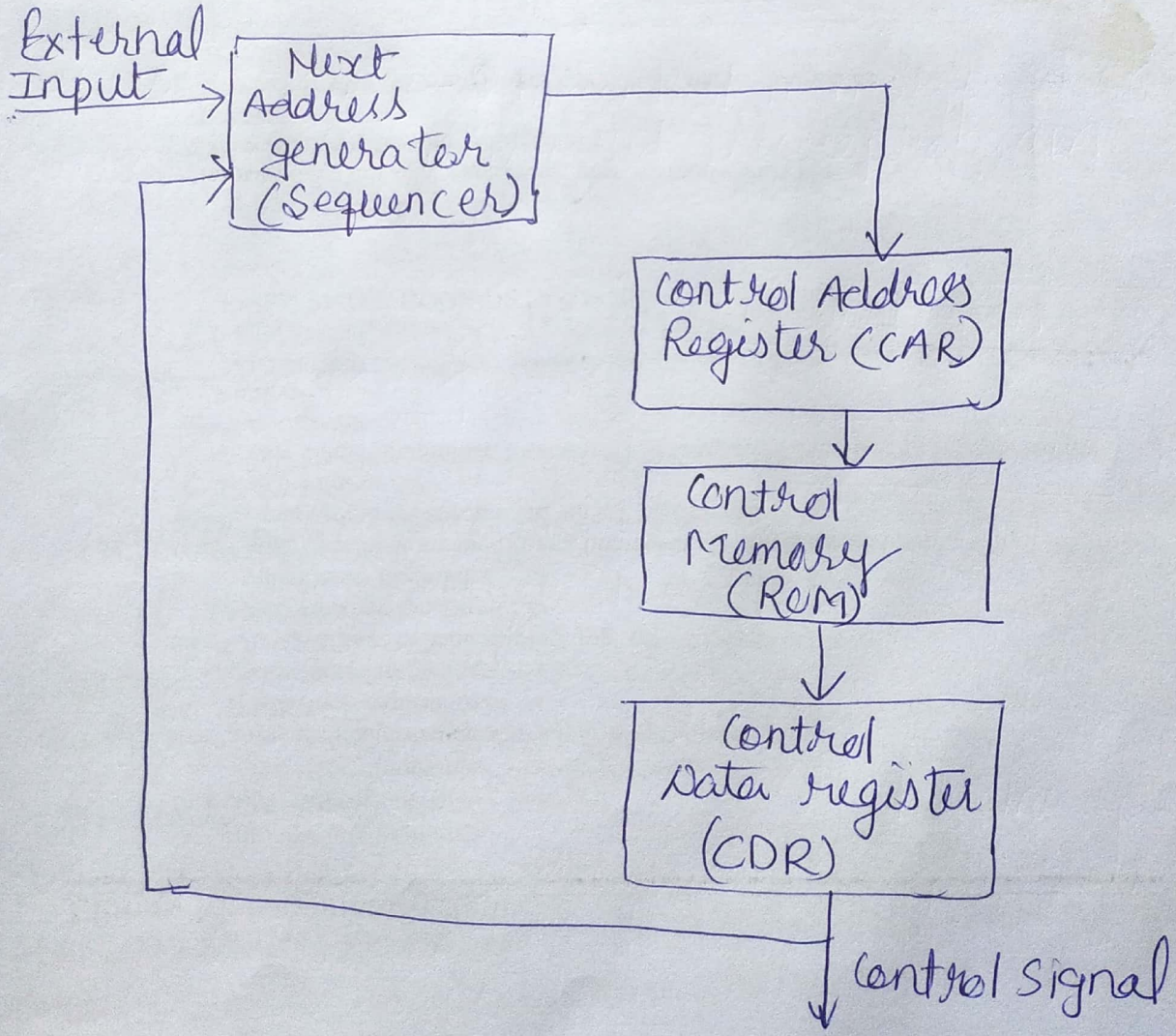
Microoperation ⇒ An elementry operation performed on data stored in registers or memory is called microoperation.

Microprogrammed control unit

⇒ When control signal are generated by the microprogram then this type of control unit is called Microprogrammed control unit

⇒ It has following components

1. Next Address generater (Sequencer)
2. Control address register (CAR)
3. Control memory (ROM)
4. Control data register (CDR)



Control memory (ROM) ⇒

⇒ The control memory hold a fixed microprogram.

⇒ The microprogram consists of microinstructions that specify various internal control signals for execution of microoperations.

Control Address register (CAR) ⇒ It holds the address of microinstruction in control memory

Control Data Register (CDR) ⇒ It holds the microinstruction read from control memory.

Sequencer ⇒ The next address generater is also called microprogram sequencer. It increment the control address register by one, load the CAR by an address from control memory, transferring an external address, or loading an initial address to start the control operations.

Advantages of Microprogrammed control unit :-

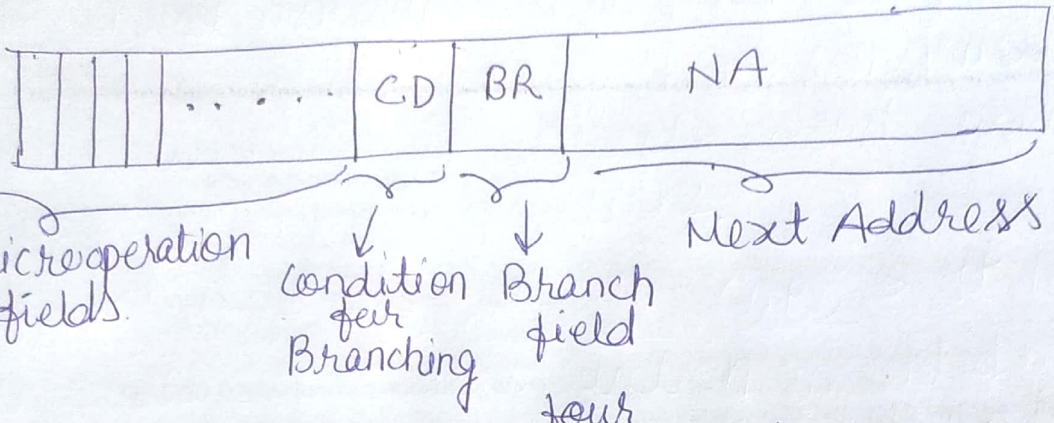
① The design of microprogrammed control unit is less complex as compared to horizontal control unit

② In it modification is easy as it involves simply changing the content of control memory.

③ It is used is CISC (complex instruction set computer) processor.

Disadvantages ⇒ It is slower than hardwired control unit. Since microinstructions are stored in control memory and fetching them takes time.

Microinstruction Format



⇒ It is divided into four fields:

① Next Address ⇒ It contains the next address or branch address of microinstruction

② BR (Branch field) ⇒ The branch field specifies the type of branch to be used.

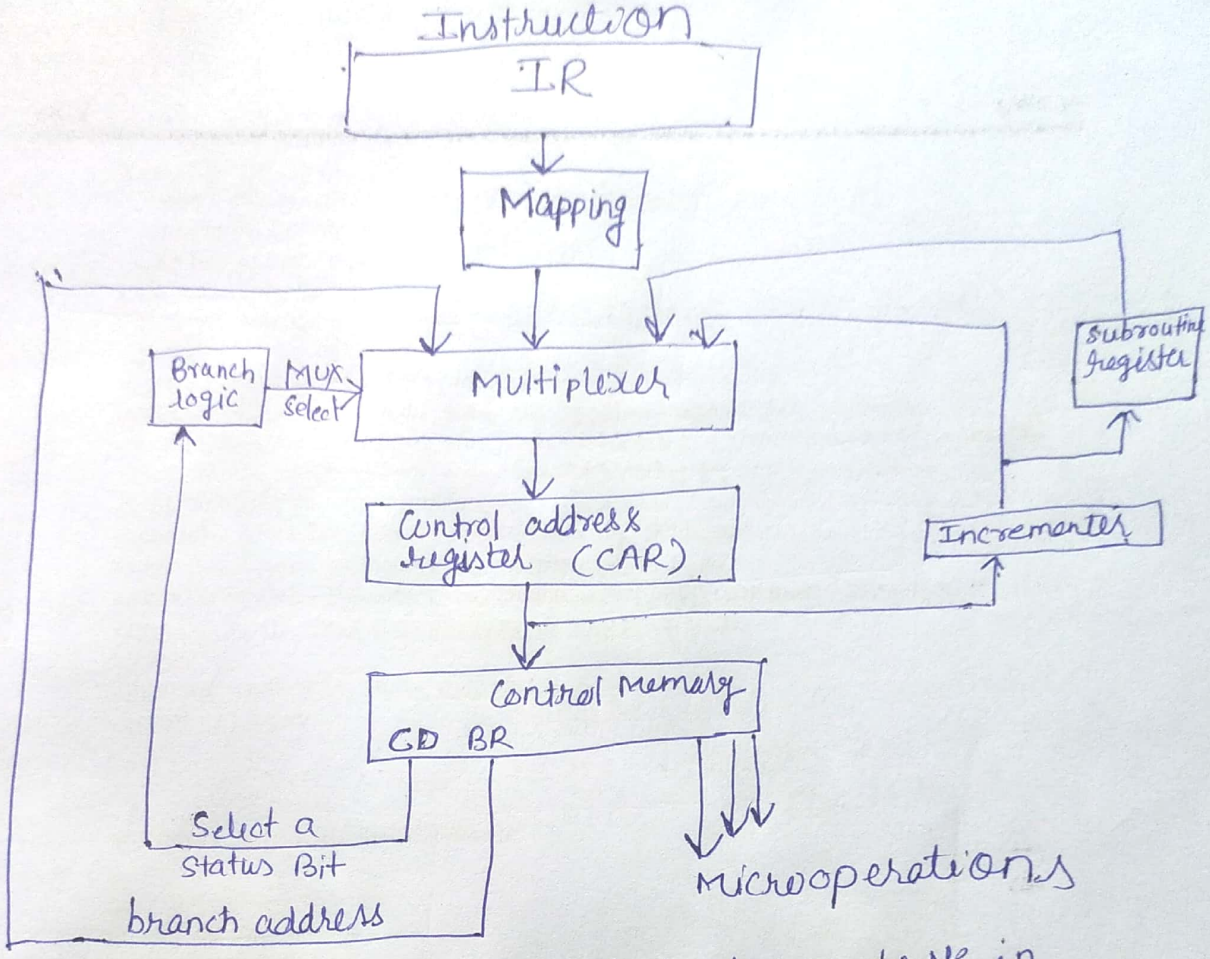
③ CD (Condition for branching) ⇒ The CD field select status bit conditions.

④ Microoperation field ⇒ It specify the different operations

Next-Address generator or Microprogram Sequencer

⇒ It determines the next address of the microinstruction.
⇒ The address of the next microinstruction can be find in several ways, depending on the sequencer inputs.
⇒ Typical functions of a microprogram sequencer are incrementing the control address register by one, loading into control address register an address from control memory, transferring external address or loading an

initial address to start control operations



⇒ The microprogram sequencer perform task in control memory to find the next address are:

1. Incrementing of the control address register
2. unconditional branch or conditional branch, depending on status bit conditions.
3. finding the address from subroutine call and return
4. A Mapping process from the bits of the instruction to an address for control memory.

Horizontal Microprogramming

⇒ Horizontal microprogramming is a microprogramming in which microprogram is a set of Horizontal microinstruction.

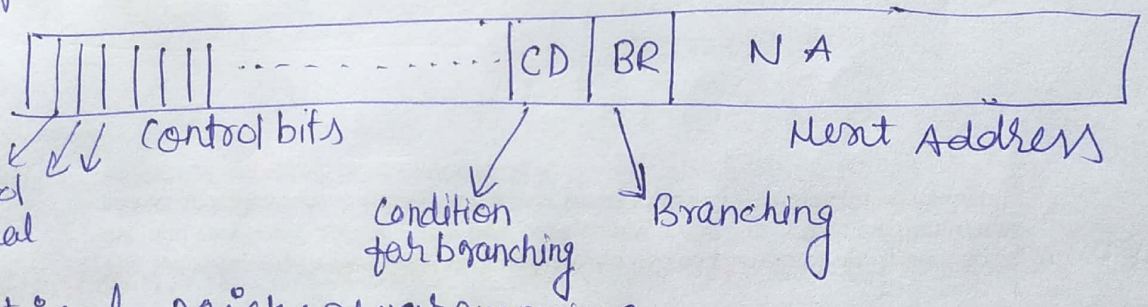
Horizontal Microinstruction :

⇒ Having an individual bit for each control signal in microinstruction format is known as a Horizontal microinstruction.

⇒ In it each bit activates one control signal, several control signals can be simultaneously generated by a single microinstruction.

⇒ The length of the microinstruction is increased

⇒ The horizontal microprogram releases faster control signals. But it required more space in control memory.



Vertical Microprogramming :

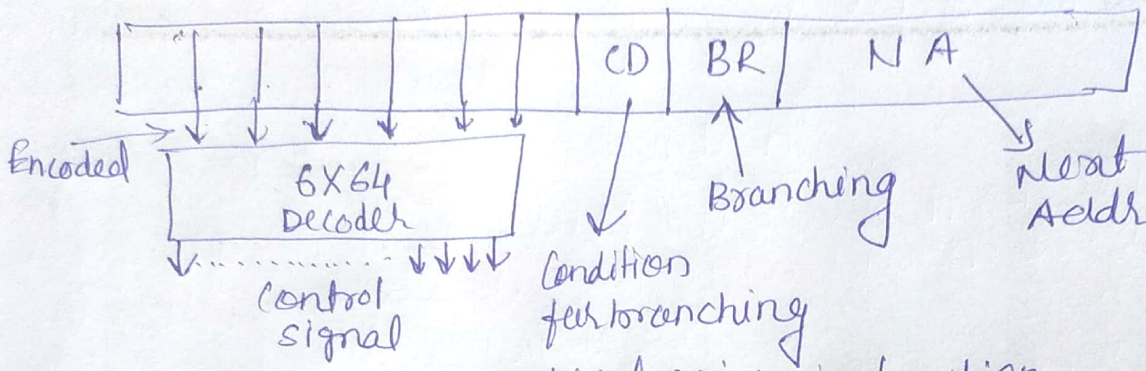
⇒ Vertical microprogramming is microprogramming in which microprogram is a set of Vertical ~~microprogram~~ microinstruction.

Vertical Microinstruction ⇒

⇒ In it the microinstruction are decoded to find the desired control signals.

⇒ This technique takes more time for generating the control signal due to the decoding time and also the more microinstruction are needed. But the overall cost is less since the microinstruction are small in size we need.

But the overall cost is less since the microinstructions are small in size.



Vertical Microinstruction

② Explain the difference between hardwired control and microprogrammed control. It is possible to have a hardwired control associated with a control memory?

Sol^{n}

Hardwired control unit	Microprogrammed control unit
① Control signals are generated by the hardware.	① Control signals are generated by microprogram
② Hardware are used to implement hardwired control unit.	② Microprogram are used to implement microprogrammed control unit.
③ After design we can not modify it	③ We can modify it by just changing the microprogram
④ It is costly due to hardware	④ It is not costly due to microprogram.
⑤ It is faster	⑤ It is slower
⑥ It is used in RISC processors.	⑥ It is used in CISC processors.
⑦ No. of instruction set are small	⑦ No. of instruction set are large.

⇒ No, Hardwired control unit does not contain a control memory.

Wide-Branch Addressing ⇒

⇒ In microprogramed control unit as the number of branches increases, generating address for each branch becomes difficult.

⇒ To handle this type of situations, the best and simple way is used programmable logic Array (PLA) to generate the required branch addresses and this way of generating the branch address is known as wide-branch-addressing.

⇒ In this method, the opcode of the instruction is translated into the starting address of the corresponding microroutine.

⇒ The opcode bits of the instruction register (IR) is connected as inputs to the PLA. Hence PLA acts as a decoder and outputs the address of the ~~dito~~ desired microroutine.

Program Control

⇒ A program control type of instruction, when executed, may change the address value in the (PC) program ~~count~~ counter register and cause the flow of control to be altered.

⇒ i.e Program control instructions specify conditions for altering the content of the program counter (PC).

⇒ Some program control instructions are

Name	Mnemonic
Branch	BR
Jump	JMP
SKIP	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

Branch and Jump ⇒ The branch and jump instructions are used interchangeably.

ex ⇒ BR Address
BR 1000

⇒ Branch and jump instructions may be conditional or unconditional.

⇒ An unconditional branch instruction causes a branch to the specified address without any conditions.

⇒ The conditional branch instruction specifies a condition such as branch if positive or branch if zero

⇒ If the condition is met, the program counter (PC) is loaded with the branch address and the next instruction is taken from this address.

⇒ If condition is not met, PC is not changed and next instruction is taken

from the next location in sequence.

SKIP ⇒ The SKIP instruction does not need an address field and is therefore a zero-address instruction

Call and return ⇒ The call and return instructions are used with subroutines.

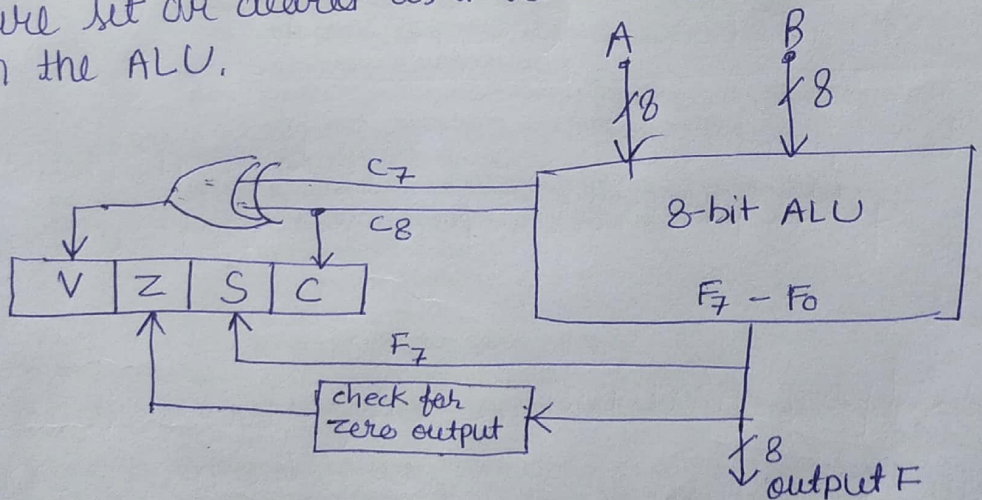
Compare and Test ⇒ The compare and test instructions do not change program sequence directly.

⇒ The compare instruction performs a subtraction between two operands. certain status bit conditions are set as a result of operation.

⇒ Similarly, the Test instruction performs the logical AND of two operands and updates certain status bits

Status Bit Conditions :- Fig below shows the block diagram of an 8-bit ALU with a 4-bit status register.

⇒ The four status bits are symbolized by C, S, Z and V. The bits are set or cleared as a result of an operation performed in the ALU.



1. Bit C (carry) is set to 1 if the end carry C_8 is 1. It is cleared to 0 if the carry is 0
2. Bit S (sign) is set to 1 if the highest-order bit F_7 is 1. It is set to 0 if the bit is 0
3. Bit Z (zero) is set to 1 if the output of the ALU contains all 0's. it is 0 otherwise
4. Bit V (overflow) is set to 1 if the XOR of last two carries is equal to 1, and cleared to 0 otherwise.

Conditional Branch instructions

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	Z=1
BNZ	Branch if not zero	Z=0
BC	Branch if carry	C=1
BNC	Branch if no carry	C=0
BP	Branch if plus	S=0
BM	Branch if minus	S=1
BV	Branch if overflow	V=1
BNV	Branch if no overflow	V=0

ex⇒ Let A = 11110000 and B = 00010100 Perform A-B and find C, S, V, and Z status bits.

Soln A: 11110000, B = 00010100

2's complement of B

= 11101011
+1
11101100

A-B = A + $\bar{B}$ + 1

A = 11110000
$\bar{B}$+1 = + 11101100
A-B = 1 11011100
↑ carry

C=1, because there is carry out of the last stage.
S=1, because the left most bit is 1.
V=0, because the last two carries are both equal to 1 and 1⊕1=0,
Z=0, because the result is not equal to 0.

(8-39)
(2) The program in a computer compares two unsigned numbers A and B by performing a subtraction A-B and updating the status bits. Let A = 01000001 and B = 10000100

ⓐ Evaluate the difference and interpret the binary result.

ⓑ Determine the values of status bits C (borrow) and Z.

Soln

$A = 01000001 = 65$

$B = 10000100 = 132$

-67

2's complement of $B = 01111011$ (one's compl)

$+1$

01111100

$A - B = A + \bar{B} + 1$

$A: 01000001$

$\bar{B}+1: +01111100$

$A - B: \quad 10111101$

$C = 1$

(65<132)

(A<B)

ⓑ C (borrow) = 1, Z = 0

BNZ (Branch if not zero)

(8.39)

Q. The program in a computer compares two signed numbers A and B by performing the subtraction A-B and updating the status bits. Let A = 01000001 and B = 10000100

ⓐ Evaluate the difference and interpret the binary result.

ⓑ Determine the value of status bits S, Z and V.

Soln ⇒

Subroutine Call and Return

⇒ A Subroutine (or Subfunction) is a self-contained sequence of instructions that performs a given computational task.

⇒ During the execution of program, a subroutine may be called to perform its function many times at various point in main program.

⇒ Each time a subroutine is called, a branch is executed to the beginning of the subroutine to start executing its set of instructions. After the subroutine has been executed, a branch is made back to the main program.

⇒ The instruction that transfers program control to a subroutine are call subroutine, Jump to subroutine, branch to subroutine.

⇒ A call subroutine instruction consists of an operation code together with an address that specifies the beginning of the subroutine. The instruction is executed by performing two operations:

① The address of the next instruction available in the program counter (PC) register (the return address) is stored in a temporary location (→ Stack) so the subroutine knows where to return.

② Control is transferred to the beginning of the subroutine.

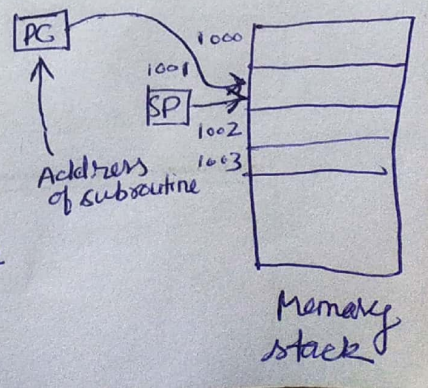
⇒ The last instruction of every subroutine commonly called return from subroutine, transfer the return address from the temporary location into the program counter.

⇒ The most efficient way is to store the return address in a memory stack, A subroutine call is implemented with the following microoperations:

SP ← SP-1 Decrement the stack pointer

M[SP] ← PC PUSH the content of PC onto the stack.

PC ← effective address Transfer control to the subroutine



If another subroutine is called by the current subroutine, the new return address is pushed into the stack, an so on. The instruction that returns from the last subroutine is implemented by the microinstruct

PC ← M[SP] Pop Stack and transfer to PC
SP ← SP+1 Increment Stack pointer.

8-32
(a) The content of the top of a memory stack is 5320. The content of the stack pointer SP is 3560. A two-word call subroutine instruction is located in memory at address 1120 followed by the address field of 6720 at location 1121. What are content of PC, SP and top of the stack:

(a). Before the call instruction is fetched from memory?
(b). After the call instruction is executed?
(c). After the return from subroutine?

(a) Before call PC, SP, Top of Stack

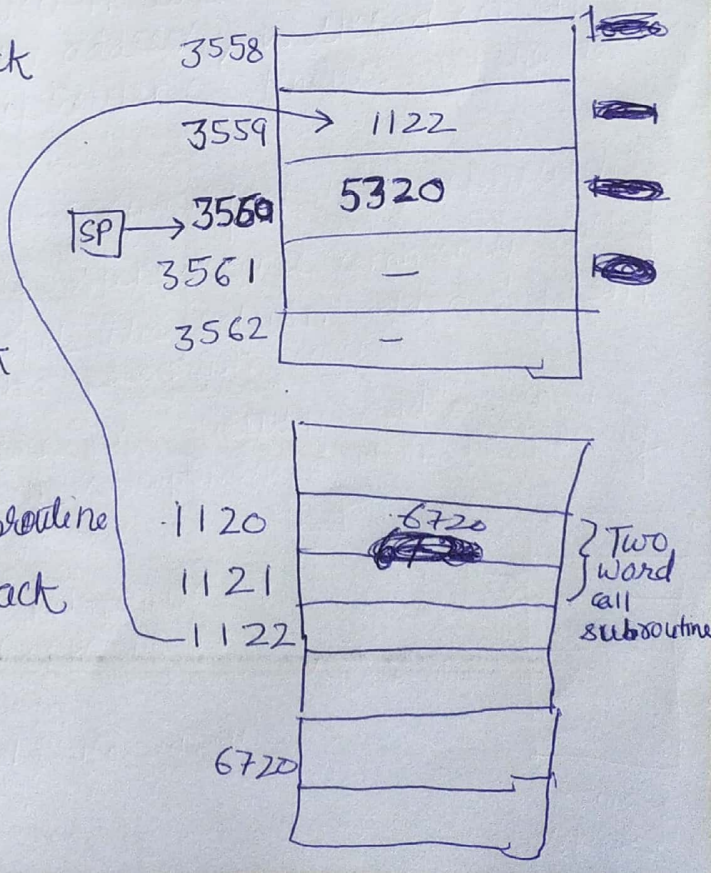
PC	SP	Top of stack
1120	3560	5320

(b) After the call:

PC	SP	Top of stack
6720	3559	1122

(c) After the return from subroutine

PC	SP	Top of stack
1122	3560	5320



2009-10

(a) (b) How many times does the control unit refer to memory when it fetches and executes an indirect addressing mode instruction if the instruction is

(a) A computational type requiring an operand from memory.

(b) a branch type.

Soln ⇒ Let one word per instruction or operand

(a) Computational type :— Total memory references will be

① Fetch instruction
② Fetch effective address
③ Fetch operand

3 Memory references.

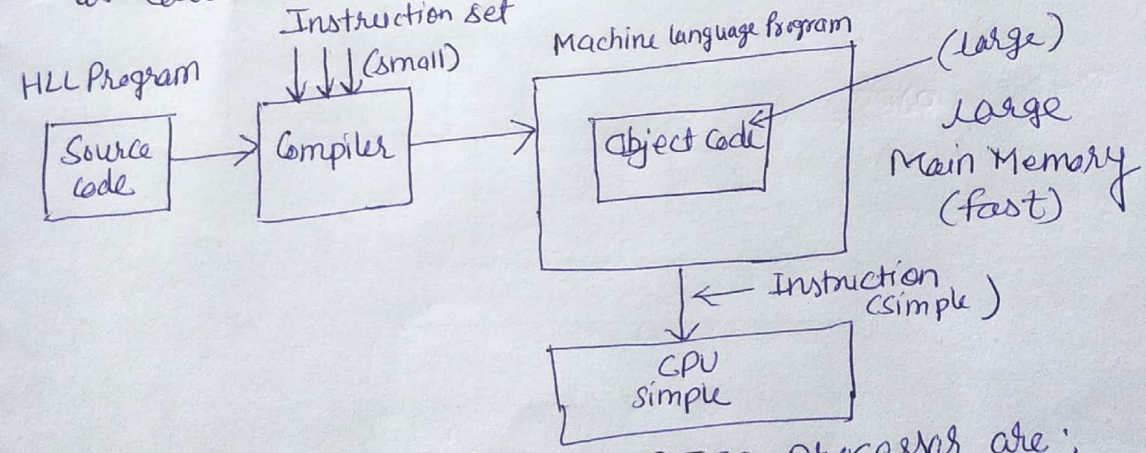
(b) Branch type :-

① Fetch instruction
② Fetch effective address and transfer to PC

2 Memory references (As here, no operand Fetch)

RISC (Reduced instruction set computer)

⇒ A computer with small number of instructions set is called reduced instruction set computer (RISC).



⇒ The main characteristic of RISC processor are :

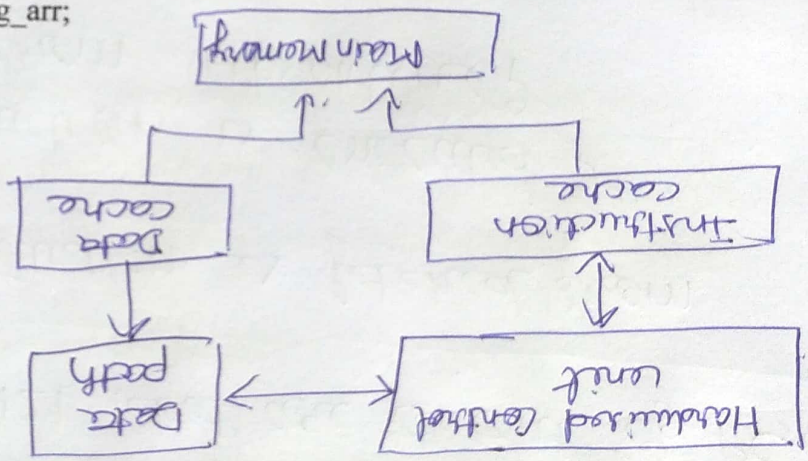
① Relatively few Instructions
② Relatively few addressing modes
③ Fixed-length easily decoded instruction format.
④ Large number of registers for storing operands.

```
        $result=mysql_query($sql);

        // Mysql_num_row is counting table row
        $count=mysql_num_rows($result);

        // If result matched $myusername and $mypassword, table row must be 1 row

        if($count>0)
        { session_regenerate_id();
$member = mysql_fetch_assoc($result);
$_SESSION['SESS_username'] = $member['username'];
$_SESSION['SESS_password'] = $member['password'];
session_write_close();
header("location:teacherlogin1.php");
exit();
}else {
//Login failed
$errmsg_arr[] = 'user name and password not found';
$errflag = true;
if($errflag) {
$_SESSION['ERRMSG_ARR'] = $errmsg_arr;
session_write_close();
header("location: teacherlogin.php");
exit();
}
}
?>
```



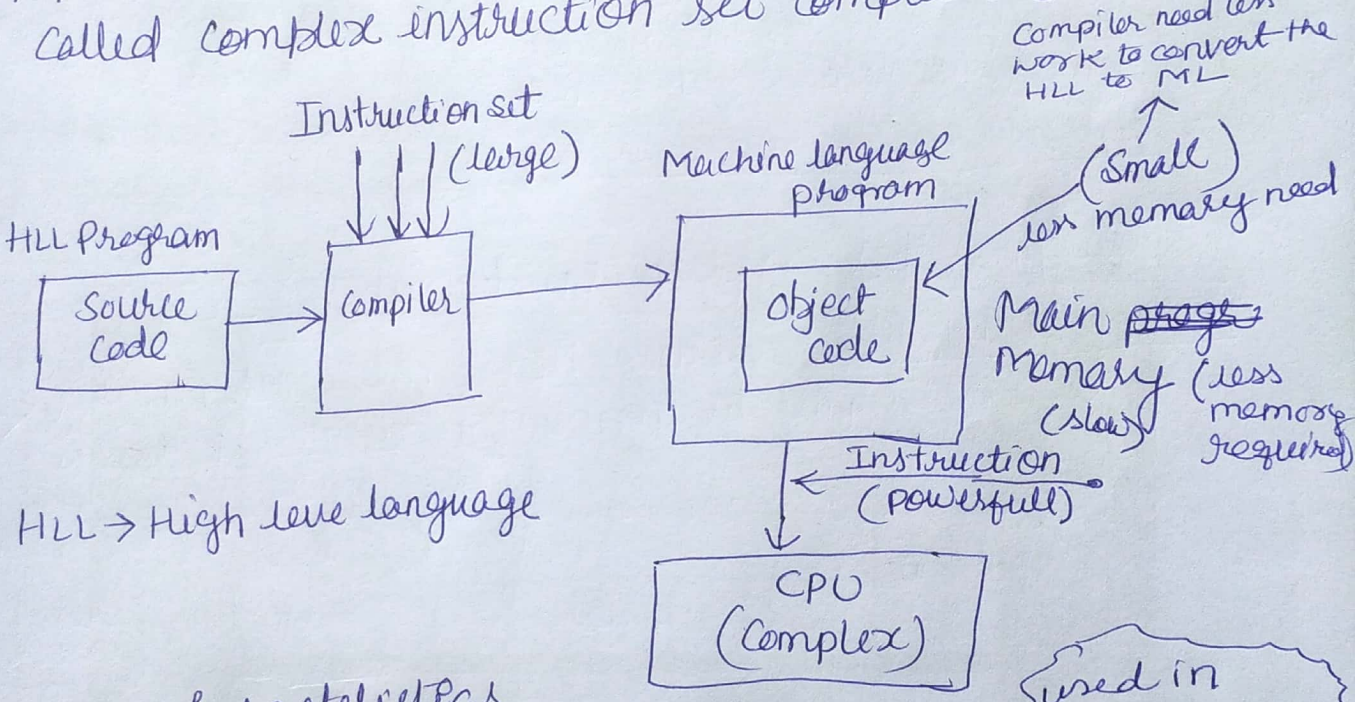
4.1.3.2 Code forSuggestion

```
<?php
session_start();
?>
<!DOCTYPE html>
<html xml:lang="EN" lang="EN" dir="ltr">
<head>
<title>SOE LIBRARY INFORMATION SYSTEM</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="imagetoolbar" content="no" />
<link rel="stylesheet" href="../styles/layout.css" type="text/css" />
<!-- Homepage Specific Elements -->
<script type="text/javascript" src="../scripts/jquery-1.4.1.min.js"></script>
<script type="text/javascript" src="../scripts/jquery-ui-
1.7.2.custom.min.js"></script>
```

⑤ Hardwired control unit is used.
⑥ Memory access limited to load and store instruction
⑦ All operation done within the registers of the CPU.
⑧ Single-cycle instruction execution.
⑨ Due to less no. of instructions it required less hardware so it's cost is less.
⑩ used in IPad, smartphones, tablets.

Complex instruction set computer (CISC)

⇒ A computer with large number of instructions is called complex instruction set computer (CISC).



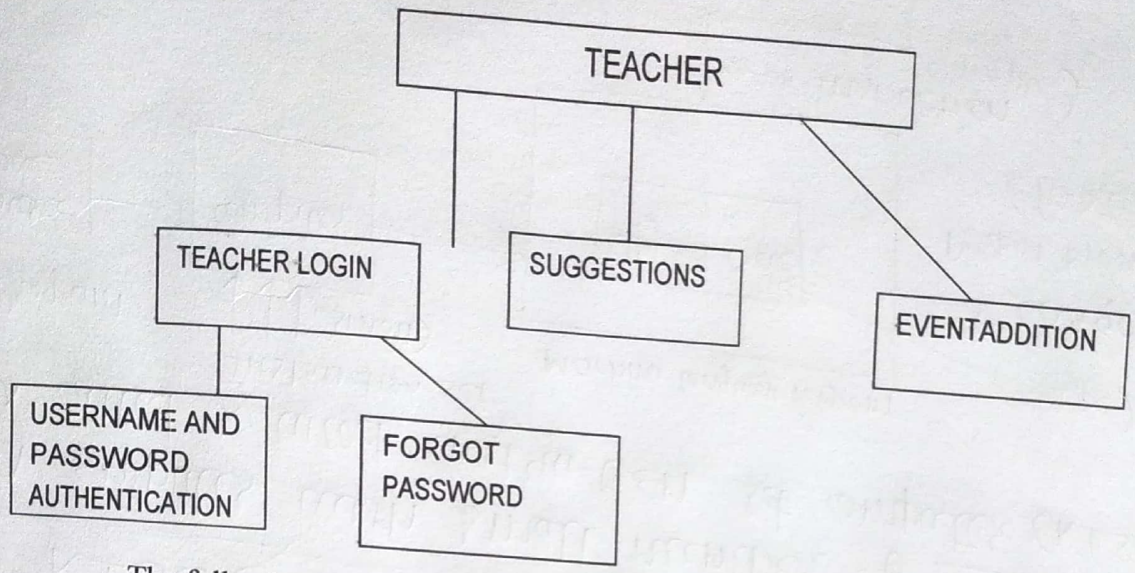
HLL → High level language

CISC characteristics

① A large number of instructions
② large number of addressing modes.
③ Variable-length instruction formats.
④ Due to large number of instruction main memory is slow.
⑤ Due to large number of instruction CPU is complex.
⑥ Some instructions that perform specialized tasks are used infrequently.
⑦ Due to complex CPU it need a lot of hardware. Due to which it is costly. ⑧ It used Microprogrammed control unit.

used in Desktop and laptop processors

- **4.1.3 Teacher Module**



The following module contains various facilities like teacher login, suggestions, and event addition. Further any teacher if at any moment forgets his/her password he/she can retrieve it from 'forgot password' option.

4.1.3.1 Code For Teacher Login

```
<?php
        session_start();
        $host="localhost"; // Host name
        $username="root"; // Mysql username
        $password=""; // Mysql password
        $db_name="library"; // Database name
        $tbl_name="teacherlogin"; // Table name

        // Connect to server and select databse.
        mysql_connect("$host", "$username", "$password")or die("cannot connect");
        mysql_select_db("$db_name")or die("cannot select DB");

        // username and password sent from form
        $myusername=$_POST['username'];
        $mypassword=$_POST['password'];
        $sql="SELECT * FROM $tbl_name WHERE username='$myusername' and
password='$mypassword'";
```

Pipelining

⇒ Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased.

⇒ Simultaneous execution of more than one instruction takes place in a pipelined processor.

Execution in a pipelined processor

⇒ Execution sequence of instructions in a pipelined processor can be visualized using a space-time diagram.

⇒ For example, consider a processor having 4 stages and let there be 2 instructions to be executed.

⇒ We can visualize the execution sequence through the following space time diagrams of Non-pipeline and pipeline

Stage/cycle	1	2	3	4	5	6	7	8
Fetch instruction	I_1				I_2			
Decode instruction		I_1				I_2		
operand Address finding			I_1				I_2	
Execute Instruction				I_1				I_2

Non-pipeline Total Time = 8 cycle

Stage/cycle	1	2	3	4	5	6	7	8	9
Fetch Instruction	I_1	I_2							
Decode Instruction		I_1	I_2						
operand Address Finding			I_1	I_2					
Execute Instruction				I_1	I_2				

Pipeline Total time = 5 cycle