Unit 3 : Control Unit

Instruction, Instruction cycle, instruction type, Instruction format, instruction format field, control unit, Implementation CU, Microprogram sequencer, execution of complete process, Risc and cisc, Pipelining Hardwired, microp microprogrammed

Horizontal Vertical

Syllabus

Control Unit: Instruction types, formats, instruction cycles and sub cycles (fetch and execute etc), micro operations, execution of a complete instruction. Program control, Reduced Instruction Set Computer, Pipelining. Hardwire and micro & programmed control; micro programmed sequencing, concept of horizontal and vertical microprogramming

Page No.
Date:

Instruction Cycle

The Basic function of computer is execution of a program. A program consists of a set of instruction.

To process and instruction

1. The process reads (fetch) an instruction from memory.
2. The processor execute the instruction.
3. The instruction execute may involved.

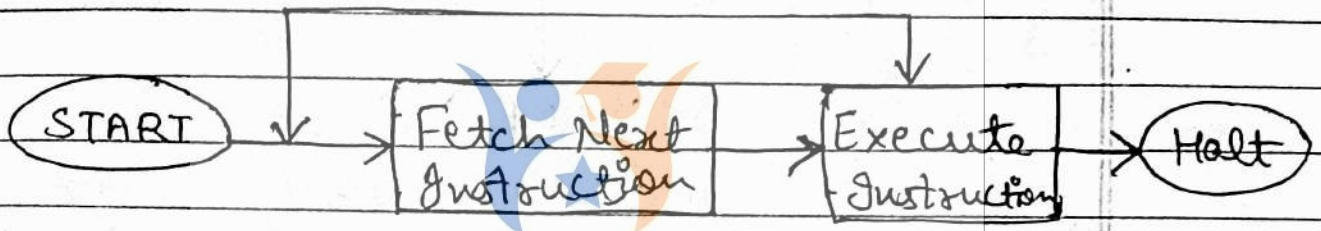


Fig: Instruction Cycle

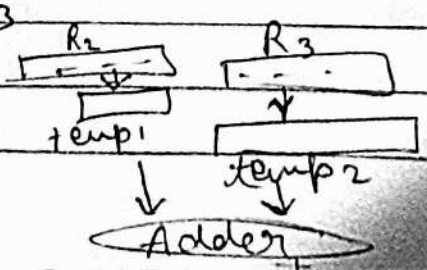
Ex: Add X, Rx

X ← X + Rx

Steps

1. Fetch the add instruction.
2. Read the content of memory location x into the processor.
3. Add Contents of x and reg Rx.
4. Write the result from the processor to memory location x.

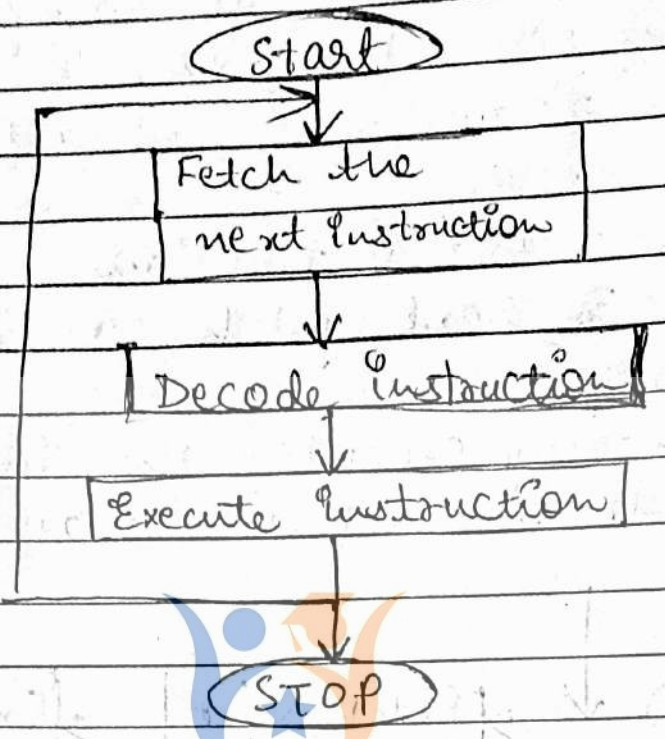
1. Load R_1 //from memory location
2. Add R_1 R_2 R_3
3.



4. Store data from processor register into a given memory location

Page No.
Date:

Instruction cycle / Instruction Subcycle



An instruction cycle involves 3 subcycle

① Fetch
② Decode
③ Execute

① Fetch : The Fetch phase reads the next instruction from memory into the CPU.

② Decode : The decode phase interprets the opcode by decoding it.

③ Execute : The execute phase performs the indicate operation.

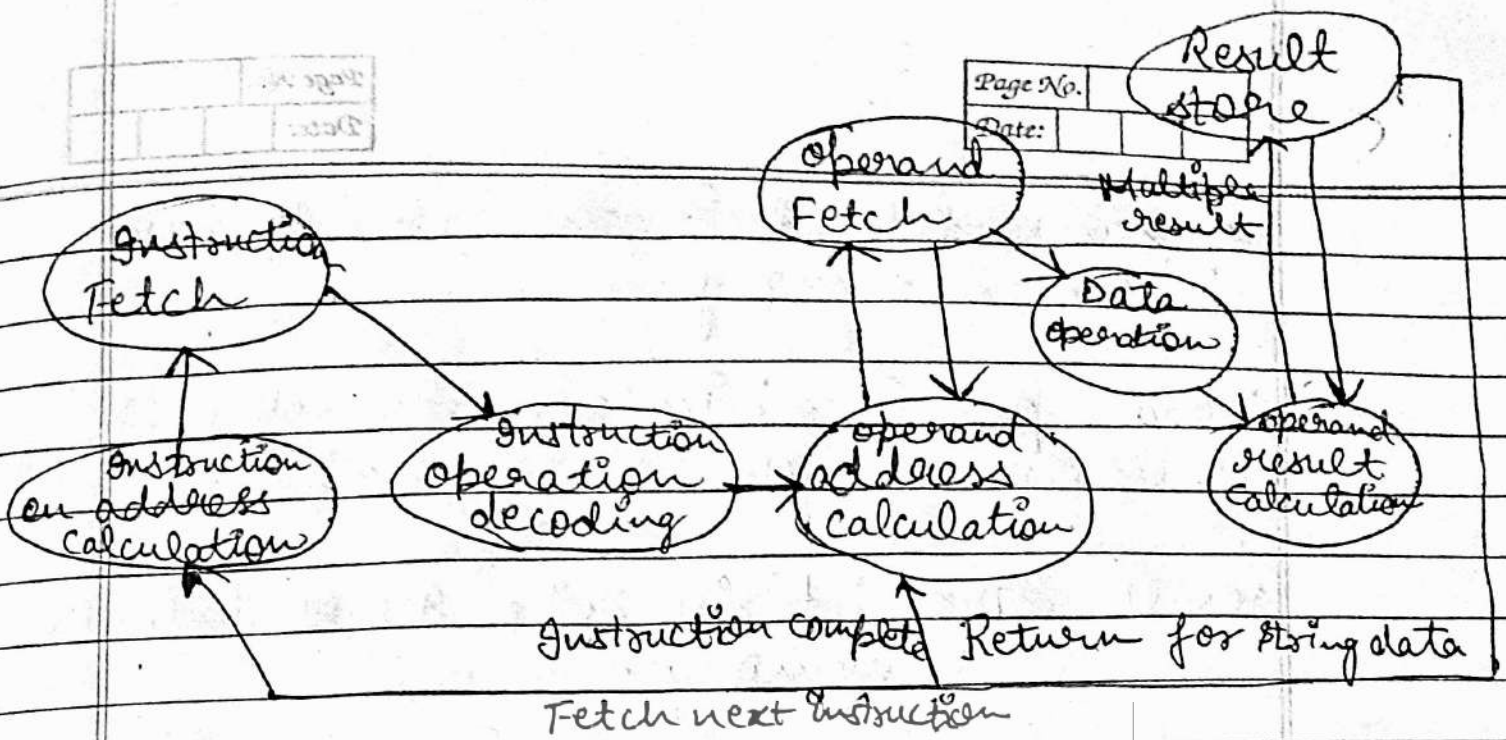


Fig: Instruction cycle state diagram

Instruction Fetch: Read instruction from memory into the processor.

Instruction operating decoding: Analyse instruction to determine type of operation to be performed and operands to be used.

Instruction address Calculation: Determine the address of next instruction to be executed.

Operand address calculation: If the operation involved reference to an operand in memory they determine the address of operand.

Page No.
Date:

operand fetch: Fetch the operand from memory.

Data operation: Perform the operation indicated in the instruction.

Result store: Write the result into the memory.

Instruction Types

A computer has set of instruction which can be classified as:

1. Data Transfer instruction
2. Arithmetic transfer instruction
3. Logical transfer instruction
4. Shift and rotate instruction
5. Program control instruction

(1) Data Transfer Instruction.

Data Transfer instruction performs the following data transfer operation

1. Data Transfer between memory and CPU registers.
2. Data Transfer between processor and I/P output device.

Instruction	Memonic	Description
Load	LD	It transfer data from specified memory location to the processor

Page No.	
Date:	

Store	ST	It transfer data from processor register to specified memory location.
Move	MOV	It transfer data b/w processor and memory
Output	OUT	It transfer data from processor or to output terminal
Push	PUSH	It transfer data from processor reg to stack memory
POP	POP	It transfer data from stack memory to processor reg.

(2) Arithmetic Instruction

Instruction	Memonic	Description
ADD	ADD	It adds the content of two operands
Subtract	SUB	It subtracts the content of two operands
Increment	INC	It adds 1 to the value stored in a reg and memory word.
Decrement	DEC	It subtract 1 to the value stored in a reg and memory word
Multiply	MUL	It multiply the contents of two operands
Divide	DIV	It divides the contents of two operands

(3) Logical Instruction

AND	AND	It logically and the individual bits of the operand.
OR	OR	It logically or the individual bits of the operand.
Exclusive OR	XOR	It logically Ex-OR of the individual bits of the operand.
Clear	CLR	It logically ~~operand~~ to be replaced by 0.
Compliment	COM	It gives the first compliment of the specified operand.

(4) Shift and Rotate instruction

Logical Shift Right	SHR	It shift the contents of specified Reg one bit. Reg to right direction and fill vacant bit with 0.
Logical Shift left	SHL	It shift the content of one bit left direction
Arithmetic shift Right	ASHR	It shift the content of specified Reg one bit position towards right and fill ~~as~~ vacant bit with previous sign bit.

Page No.
Date:

Arithmetic Shift left ASHL It shift of the content of specified Reg bit.

(5) Program Control Instruction

Branch BR It transfer program controlled to the specified address.

Jump JMP It transfer programs controlled to specified address

Skip SKP It skip the next instruction

Call CALL It save the addr. of next instruction in the stack and transfer to the program control to ~~skip~~ specified add

Return RET It read the add from the Top of stack and transfer the program control to the read add.

Compare CMP It performs a subtraction between two operands but the result of the operation is not stored

Test TST It performs the logical AND of two operands and updates certain status bits without storing the result or changing the operands.

Page No.
Date:

Instruction Format: Each instruction of the CPU contains specific information field which are required to execute it. These information fields of instruction.

(1) Operation code: The operation code field of the instruction specifies the operation to be performed.
(2) Source / Destination operand: The source / destination operand field directly specify the source / destination operand for the instruction.
(3) Source operand Address: It specifies address of the source operand.
(4) Destination operand Address: It specifies the address of the destination operand.
(5) Next instruction Address: The next instruction address tells the CPU from where to fetch the next instruction after complete of execution of current instruction.

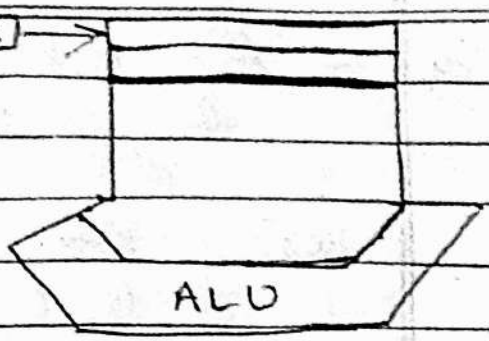
Instruction format Type

(1) Zero address instruction
(2) One address instruction
(3) Two address instruction
(4) Third address instruction.

Convert given expression into reverse polish notation (postfix form) then proceed

Page No.
Date:

Zero address instruction



Condition
* Implicit operand on stack
* No register
* Operand in ALU instruction or in push and pop.

Eg: C = A + B using zero address

```
Push A ;   TOS ← A
Push B ;   TOS ← B
Add    ;   TOS ← A + B
POP C  ;   M[C] ← TOS
```

zero address instruction include only operation code without operands where operands are used with stack op^n PUSH and POP.

opcode

Three address instruction

23 bit	6 bit	5 bit	5 bit	5 bit
	opcode	Rdest	RSrc	RSrc

Instruction
Add DEST SRC, SRC2 : M(DEST) = SRC1 + SRC2
SUB DEST SRC, SRC2 : M(DEST) = SRC1 + SRC2
MUL DEST SRC, SRC2 : M(DEST) = SRC1 + SRC2

Three address instruction include three operation along with operation code, two for source and for destination.

Page No.
Date:

Eg: $X = (A + B * C) / (D - E / F)$ to evaluate at 3-address instruction.

Instruction	Comments
MUL R_1 B, C	$R_1 \leftarrow M[B] * M[C]$
ADD R_2 A, R_1	$R_2 \leftarrow [A] + R_1$
DIV R_3 E, F	$R_3 \leftarrow M[E] / M[F]$
SUB R_1 D, R_3	$R_1 \leftarrow M[D] - R_3$
DIV X, R_2, R_1	$M[X] \leftarrow R_2 / R_1$

Two Address Format

opcode	DES	S

Two address instruction include two operation along with op code one for source and one for both source and destination address.

Eg: $X = (A + B * C) / (D - E / F)$ using two address instruction format

Instruction	Comments
LOAD DEST SRC	M(DEST) = SRC
ADD DEST SRC	M(DEST) = DEST + SRC
SUB DEST SRC	M(DEST) = (DEST) - SRC

Page No.
Date:

Eg: $A = B + C * D - E + F + A$

Instruction	Comments
Load T, C	T = C
MUL T, D	T ← C * D
ADD B, T	T ← B + C * D
SUB T, E	T ← T − E
ADD T, F	T ← T + F
Add T, A	T ← T + A

One Address Instruction (use an implied accumulator register for data manipulation)

One address instruction include only one operand along with operation part, which used for both source and destination address.

[opcode | reset | SRC]

Sample Instruction are

Instruction	Comments
~~LOAD~~ Load ADDR	Accum = [ADDR]
Store ADDR	M[ACCUM] = Accum

Eg: $A = B + C * D - E + F + A$

Instruction	Comments
Load C	A ← C
MUL D	A ← A * D
ADD B	A ← B + A
SUB E	A ← C * D + B − E
Add F	A ← C * D + B − E + F
Add A	A ← C * D + B − E + F + A
Store A	A ← [Ac]

Micro operation

A micro operation is an elementary op^{n} perform with the data store in register there are four categories:

1. Register transfer micro operation
2. Arithmetic micro operation
3. Logic micro operation
4. Shift micro operation

(1) Register transfer micro operation
Binary information one register to another register.

$R_1 \leftarrow R_2$

(2) Arithmetic Transfer Micro operation
Perform Arithmetic op^{n} on numeric data store in register.
Eg: Addition $R_3 \leftarrow R_1 + R_2$
Sub $R_3 \leftarrow R_1 - R_2$

(3) Logic micro operation
Perform bit manipulation operation on non-numeric data stored in register.
Eg: AND, OR, NOR.

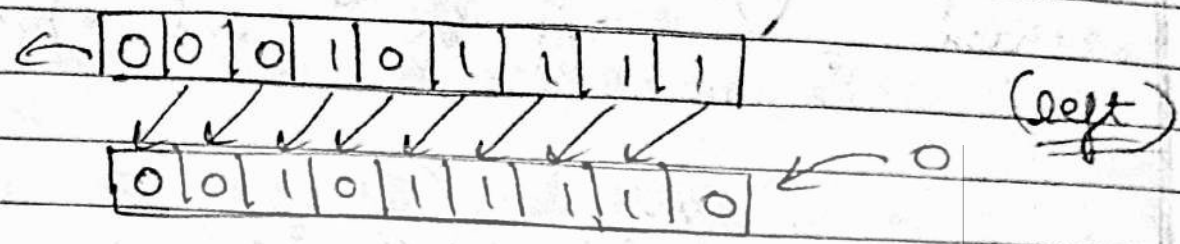
(4) Shift micro-operation
Perform shift micro-op^{n} on data stored in register. There are 3 types of

Page No.
Date:

shift-micro op" :

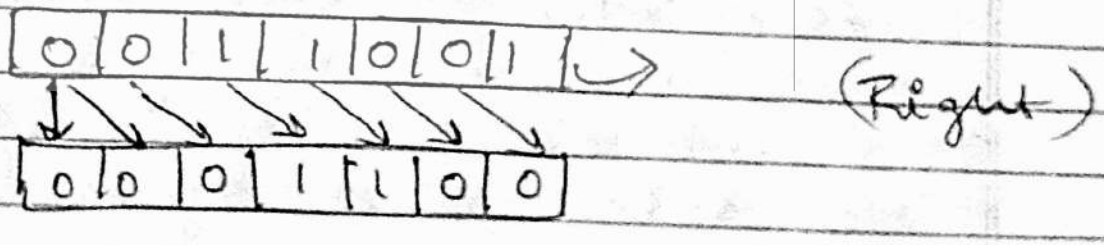
(a) Logical shift
(b) Circular shift
(c) Arithmetic shift

(a) Logical shift



(b) Circular shift

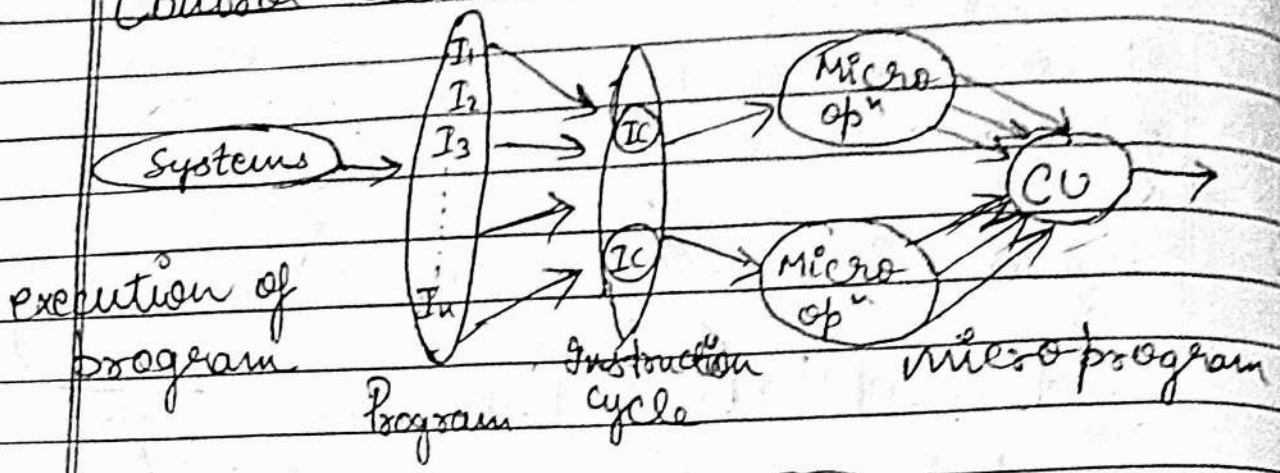
(c) Arithmetic shift



Micro operation, micro program, micro code

Page No.
Date:

Control Unit



control Unit → H/W circuit

Control Unit generates control signals based on instruction which are directly realized by H/W circuit.

System functionality is execution of programs. Program means sequence of instruction along with data. Program execution is represented by instruction cycle. Instruction cycle consists of fetch cycle and execution cycle.

Each subcycle of instruction cycle consist their own micro program.

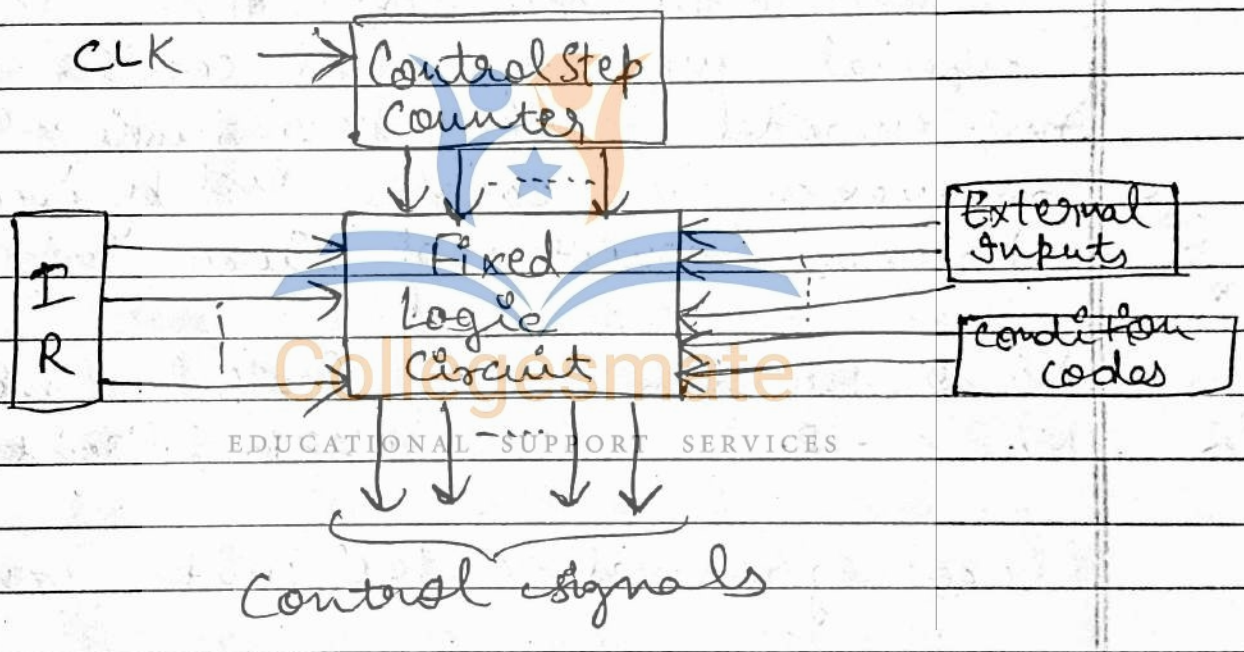
Microprogram consist of sequence of micro op^n which represent to register to register transfer op^n. It is also called as micro instruction or control word.

Page No.
Date:

⇒ Control signal generated based on micro-operation.
⇒ Control signal are directly realized on H/w circuit.

Control unit implement in 2 ways
① Hardwired control unit
② Microprogrammed control unit ← Horizontal, Vertical

Hardwired control unit



Hardwired control unit is implemented as a sequential logic circuit or finite state machine that generates a specific sequence of control signals.
In this design the control signal are directly realized on hardwired circuit.
Control signal are expressed in terms of SOP expression.
It is the fastest CU Design. It is used

in the real time application.
Even a minor modification required redesign and reconnection of the circuit. Hence it is not flexible. Therefore it is not suitable for design and testing environment.

Microprogrammed Control Unit

In microprogrammed control unit, micro-instructions are stored in a special memory called control memory. Implemented using programming approach sequences is carried out by executing a program consisting of micro-instruction.

MAR - Memory Address Register

Micro Instruction: MAR ← R3 — Control signals: MARin, R3out

Microprogrammed control unit format

Branch condition	Flags	Control signals	Control Address

Branch condition → Bz - Branch on zero, BNz - Branch on Non-zero, BC - Branch Carry, BOV, BNV

Flags → System supported Flags: C - Carry, Z - zero, S - Sign, P - Parity, OV - overflow

Control signals → Types of operation: Encoded (Vertical), Decoded (Horizontal)

Control Address → next Micro operation

In this design control memory is used to store the micro program the control memory is a permanent memory.
Eg: ROM
In the control memory the control words are stored. The format of the control words is Branch Condition, flags, control signals, control address.
Based on the way of storing the control signal in the control words we can classify the control words into 2 types:

1. Horizontal control word
2. Vertical control word

① Horizontal control word

It represent the control signal in the form of decoding patterns. One bit per control signal. When the control unit support with horizontal control word then the corresponding control unit is called as horizontal microprogram control unit.

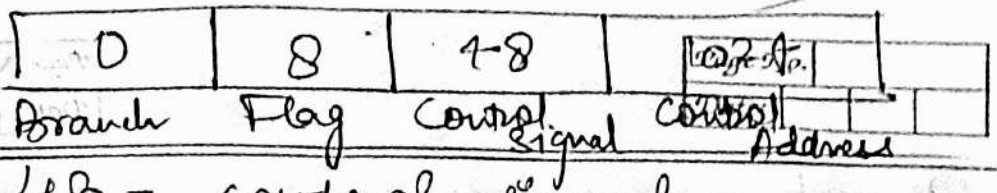
* It support with decode pattern of control signal.
* It support longer control words.
* It is used in the parallel processing application.
* It is a little bit flexible the hardwire control unit.

(2) Vertical control unit

In this format the control signals are stored in encoded binary pattern when the control unit support with control word the corresponding control unit is called as vertical / microprogrammed control unit.

* It supports encoded binary patterns of control signal and control signals required bit - $\log_2 n$.
* It is more flexible than horizontal and hardwire.
* It support smallest control word.
* There is a no parallelisms in the vertical control unit.

Ques: Consider a control unit. It support with 48 control signals and 8 flag ~~contains~~ conditions. It also support with 1024 control word memory what is the size of control word in bits and control memory in bytes using horizontal programming.



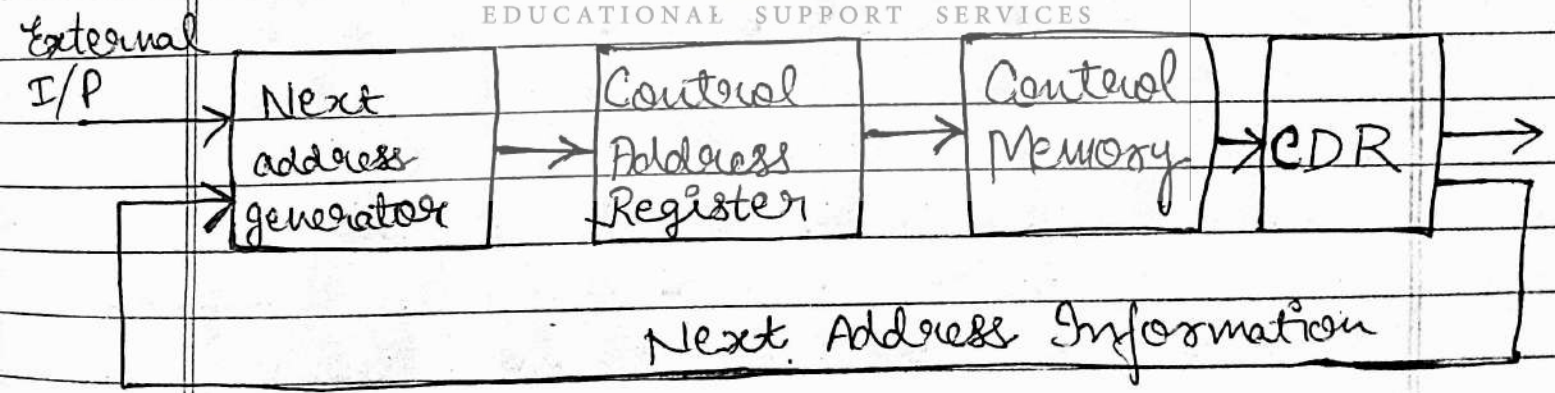
⇒ 48 - control signal
8 - Flag conditions
1024 - control word memory

Flag + $\log_2$ no. of flag

$8 + \log_2 8 = \log_2 (2)^3 = 3$ bits

Branch is not given = 0
Size = 1024 = 2^{10} = 10 bits
Control word = 0 + 3 + 48 + 10
= 61 bits

Control Memory = $\frac{1024 \times 61}{8}$
= 7808 bytes



Page No.
Date:

Microprogram Sequences

The basic components of microprogrammed control unit are the control memory and the circuits that select the next address.

The subunit of microprogrammed CU which presents an address to the control memory is called microprogram sequencer.

The next address logic of the sequencer determines the specific address source to be loaded into the (Control Address Register)

Two important factor that must be considered while designing the micro instruction sequencer.

① The size of the micro-instruction.
② The address generation time.

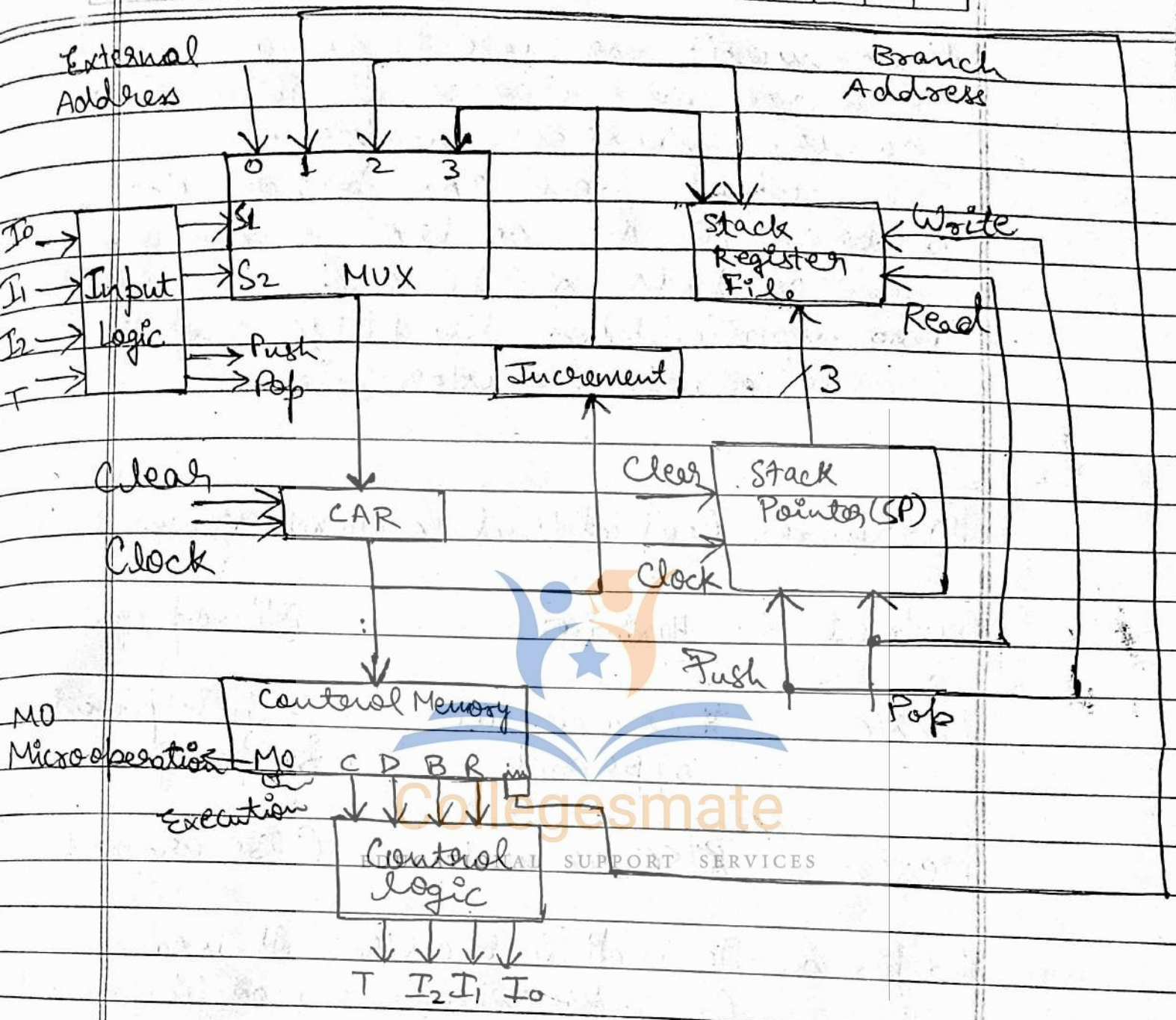


Fig: Typical Micro program Sequencer Organization

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

Page No.
Date:

Here multiplexer selects an address from 4 sources and routes it into a control address register.
The output from CAR provides the address for the control memory.
The contents of CAR are incremented and applied to the MUX and to the stack register file.

Hardwired Control Unit Vs Microprogrammed CU

Key Point	Hardwired	Microprogrammed
Basic	It is a circuit approach	It is implemented by programming
Design	RISC are used	CISC are used
Modification	It is difficult as the CU is Hardwired. Modifying it will required the change in H/W	It is easy to modification and it is change only the code.
Instructions	It works well for simple instructions	It works well for complex instructions
Control Memory	No control memory	Control memory is required
Execution Speed	Faster execution	Comparatively slow

Page No.
Date:

RISC (Reduce Instruction Set Computer) is used in portable devices due to its power efficiency.

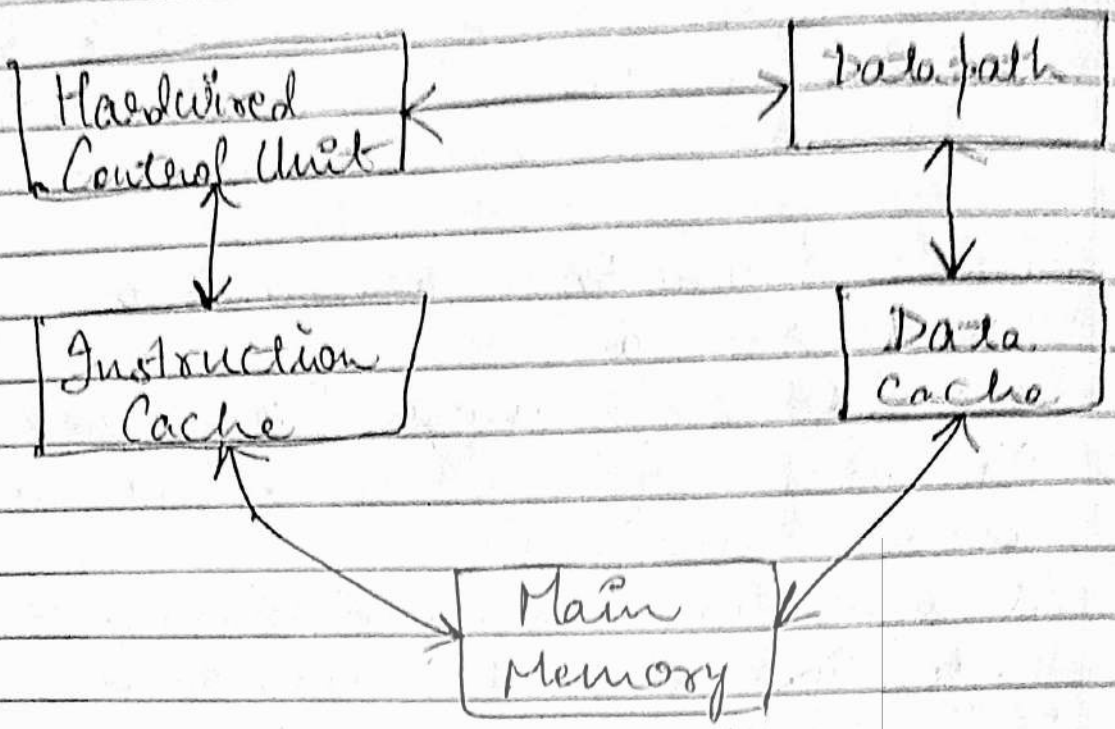
* Simple instructions are used and few instructions are used.
* Few addressing modes are used.
* Memory access limited to load and store instruction.
* All operation done within the register of CPU.
* Single cycle instruction execution.
* Fixed length easily decoded.

CISC (Complex Instruction Set Computer)

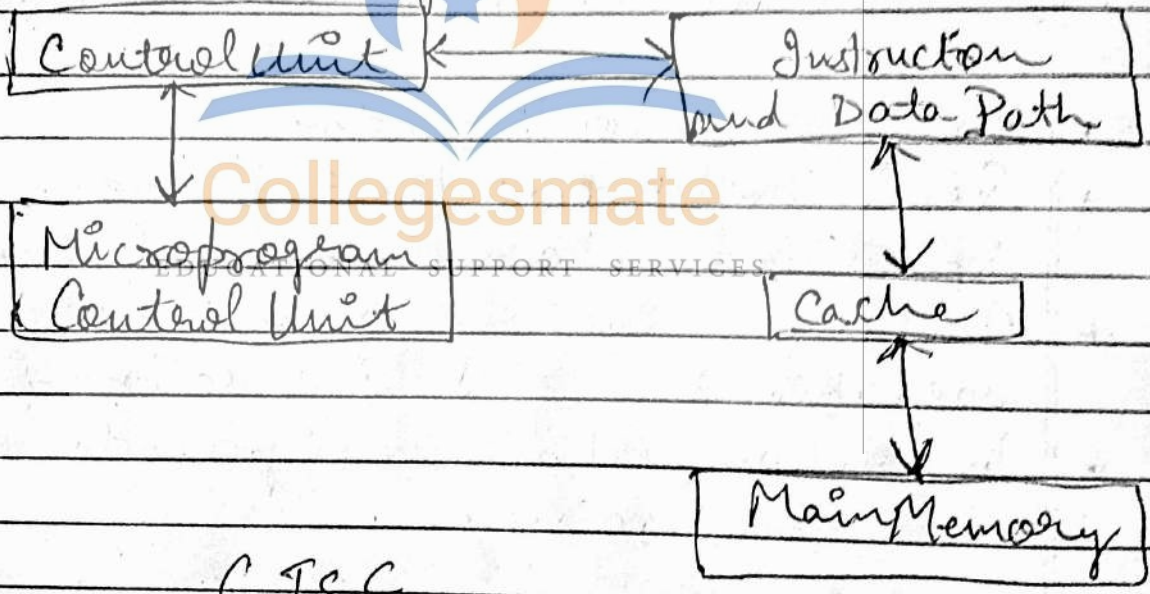
* Large no of instruction.
* Some instruction that performed specified tasks are used infrequently.
* A large variety of addressing modes variable length instruction formats.
* Instruction that manipulate operands in memory.

Architecture of RISC

It uses highly optimized set of instructions.
It is used in tablets, mobiles/smartphones, portable devices.



Architecture of CISC



CISC

Page No.
Date:

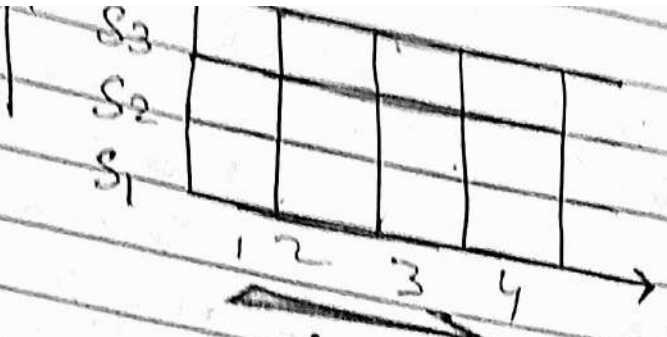
Horizontal Vs Vertical Microprogrammed

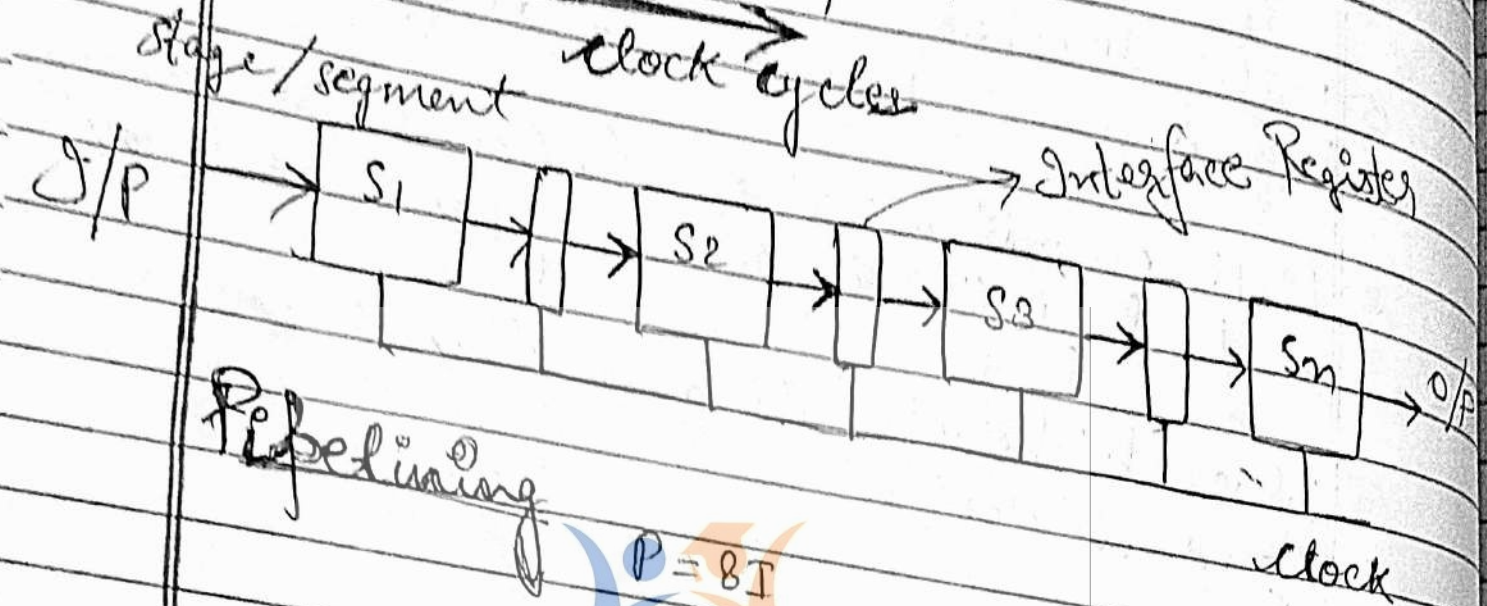
Horizontal	Vertical
① Long format used of horizontal	① Short format are used.
② Ability to express a high degree of parallism.	② Limited ability to express parallel.
③ Little encoding of the control information	③ Considerable encoding of the control information
④ When higher operating speed is design.	④ When slower operating speed is design.

Pipelining

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

Simultaneous execution of more than one instruction takes place in pipelined processor. In pipline multiple instruction are overlapped in execution.





Pipelining

P = 8I

or

Load = 8

$I_1 - I_8$

Non Pipelining

5 stage ① IF
② ID
③ Ex
④ MEM
⑤ WB
+ Write back

1 clock = 1 stage cycle to complete used

If 5 stage are used

5 CC × BI = 40 CC
non pipline to be executed

number of stage = K
number of instruct = n

n → (n-1)

CPI ≃ 1 clock per instruction

K + (n-1)
= 5 + (8-1)
= 5 + 7 = (12)//

Execution of a complete instruction

Add (R3), R1

Step	Action
1	PCout, MARin, Read, Select 4, Add, Zin
2	Zout, PCin, Yin, WMFC
3	MDRout, IRin
4	R3out, MARin, Read
5	R1out, Yin, WMFC
6	MDRout, Select Y, Add, Zin
7	Zout, R1in, End