UNIT-2

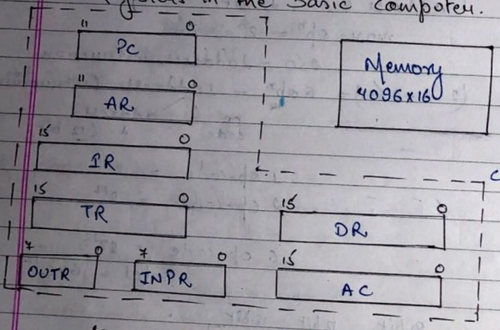
Date: 26|9|18
Page:

CONTROL UNIT

Computer Registers

Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time. The control reads an instruction from a specific address in memory and executes it. It then continuously reading the next instruction in sequence & executes it and so on.

Registers in the Basic computer.

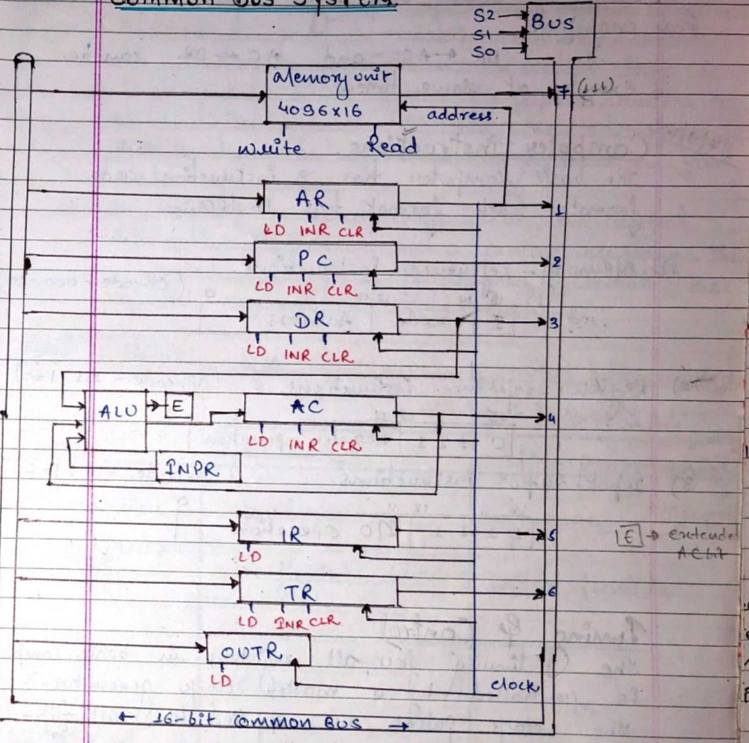


List of Registers.

DR	16	Data Register.	Holds memory operand.
AR	12	Address Reg.	Holds address for memory.
AC	16	Accumulator	Processor register.
IR	16	Instruction Reg.	Holds instruction code.
PC	12	Program counter	Hold address of instructions
TR	16	Temporary reg.	Holds temporary data
INPR	8	Input register.	Holds input character
OUTR	8	Output register.	Holds o/p character.

Date:
Page:

Common Bus System



The content of any register can be applied onto bus and an operation can be performed in the adder and logic circuit during same clock cycle. The clock transition at the end of cycle transfer the content of the bus into the destination register. and the

9100 ← indirect memory
1001 0100

Output of adder and logic circuit into AC.
For example,
DR ← AC and AC ← DR can be executed at same time.

27/9/18

Computer Instructions.

The basic computer has 3 instruction code formats. Each format has 16 bits.

1) Memory-reference instructions. (OP-code = 000 to 110)

15	14 – 12	11 – 0
I	opcode	Address

0xxx
0000
15=0 14 13 12 =0 =0 =0 ← direct memory

2) Register-reference instructions. (op-code = 111, I = 0)

15	12	11 – 0
0 1 1 1		Register operation

11th bit = 1 = clr AC
10th bit = 1 = clear E
7400
0111 0100 0000 0000
reg. ref.
0111 = 7 → 7809 clear register reference

3) Input-output instructions. (op-code = 111, I = 1)

15 – 12	11 – 0
1 1 1 1	I/O operation

op-codes
1111 = F
F800
i/p o/p

Timing & Control (Hardwired)

The timing for all registers in basic computer is controlled by a master clock generator. The clock pulses are applied to all flip-flops and registers in the system, including the flip-flop and registers in control unit. The clock pulses donot change the state of a register unless the register is enabled by a control signal. The control signals are generated in control unit.

Date: 27/9/18

difference b/w them = ?

There are two types of control organizations – Hardwired and microprogrammed control.
→ In Hardwired organisation, control logic is implemented with gates, flip-flops, decoders & other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation.
→ In microprogrammed organisation, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of micro-operations.

28/9/18

Difference b/w Hardwired & micro-programmed.

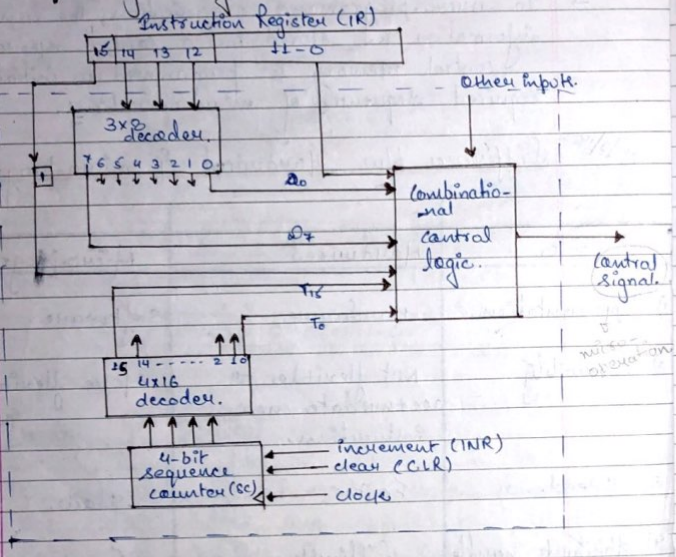
Attributes.	Hardwired	Micro-programmed
1) Implementation.	Hardware	Software.
2) Flexibility	Not flexible to accomodate new instruction.	More flexible.
3) Speed.	Fast	Slow.
4) Ability to handle complex instruction set.	difficult.	Easy.
5) Design Process.	Complicated.	Systematic.
6) Memory	No memory used.	Control memory is

Scanned with CamScanner

Date: 28/9/18
Page:

		used.
7) Applications.	Mostly RISC processor, intel 8085 motorola 6802	Main frames, CISC, intel 8080, motorola 68000

Block diagram of Hardwired control. (control unit of basic computer)

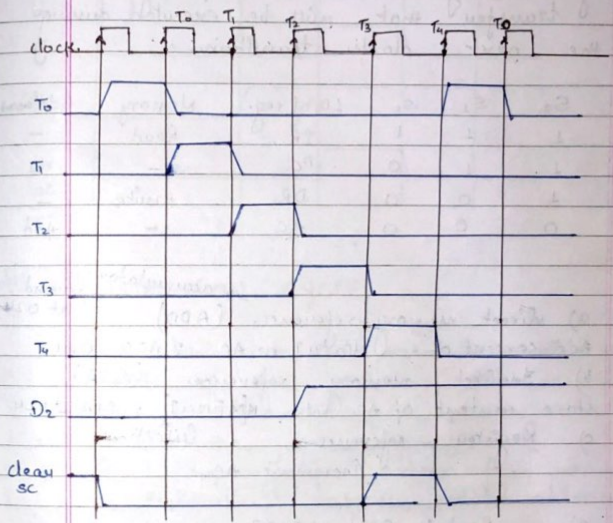


The sequence counter can be incremented or cleared synchronously. most of the time the counter is incremented to provide the sequence of timing signals out of 4x16 decoder. Once in a while the counter is cleared to zero. for the next active timing signals to be T_0

Date:
Page:

For example, consider the case where sequence counter is incremented to provide timing signals T_0, T_1, T_2, T_3 and T_4 in sequence. At time T_4, SC is cleared to zero if decoder output D_3 is active.

D_3T_4 : $SC \leftarrow 0$



Que. Consider the instruction format of basic computer for each of the following 16 bit instruction explain what instruction is going to perform -

a) $\overset{1}{0\underline{001}}$ $\overset{0}{0000}$ $\overset{2}{0010}$ $\overset{4}{0100}$ ← direct memory.

b) $\underbrace{1011}_{11=B}$ $\underset{1}{0001}$ $\underset{2}{0010}$ $\underset{4}{0100}$

c) → Reg. $0\underset{7}{\underline{111}}$ $\underset{0}{0000}$ $\underset{2}{0010}$ $\underset{0}{0000}$

Ques. Draw the timing diagram assuming that sequence counter is cleared to zero at time T_3 if control signal C_7 is active. with the clock. transition. C_7T_3 : $SC \leftarrow 0$ associated with T_1

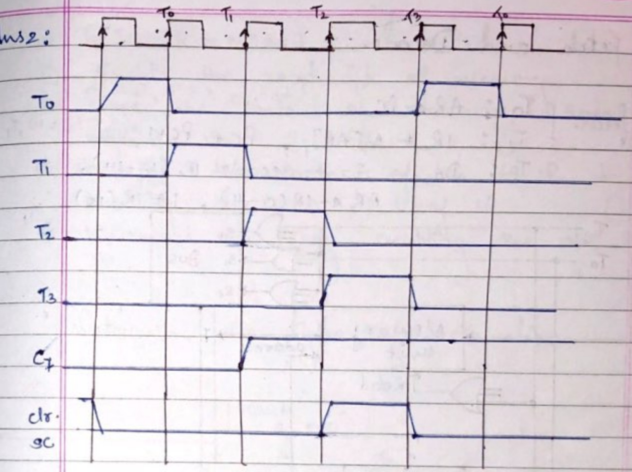
Ques. Following control i/p are active in the bus system for each case specify the register transfer that will be executed during the next clock transition.

	S_2	S_1	S_0	LD of reg.	Memory	Adder.
a)	1	1	1	IR.	Read	–
b)	1	1	0	PC	–	–
c)	1	0	0	DR	write	–
d)	0	0	0	AC	–	Add

Ans 1. a) Direct memory-reference. (ADD) → accumulator + operand at 024.
Add content of M[024] to AC ; ADD 024
b). Indirect memory reference (STA)
store content of AC into M[M[124]] ; STA I 124
c) Register - reference. (INC)
Increment AC.

Ans 2. a) ~~Read : IR ← M[AR]~~
memory read to bus. and load to IR.
IR ← M[AR]
b) TR to bus and load to PC.
PC ← TR
c) AC to bus, write to memory & load to DR.
M[AR] ← AC , DR ← M[AR]
d) Add DR or INPR to AC. , load to AC
AC ← AC+DR or AC ← AC+INPR

Ans2:



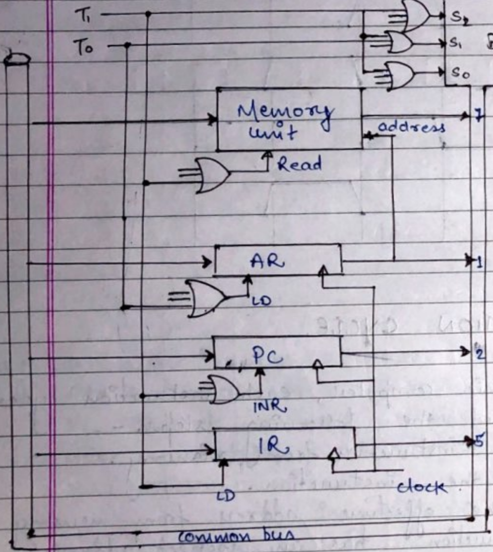
3/10/18 INSTRUCTION CYCLE.

In the basic computer, each instruction cycle consists of the following steps.-
1) Fetch an instruction from memory.
2) Decode the instruction.
3) Read the effective address from memory if the instruction has an indirect address.
4) Execute the instruction.

Upon the completion of step 4, the control goes back to step 1. to fetch, decode & execute the next instruction.

Fetch and Decode.

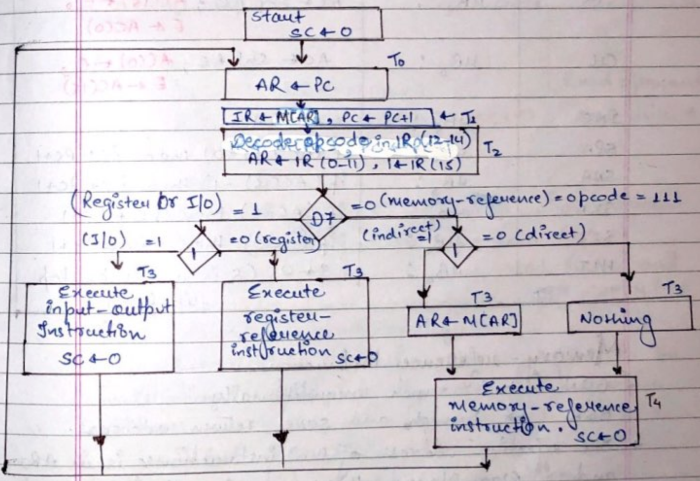
fetch. [T_0 : AR ← PC   ($S_0 S_1 S_2$ = 010, T_0=1)
T_1 : IR ← M[AR] , PC ← PC+1 ($S_0 S_1 S_2$ = 111, T_1 = 1)
T_2 : D_0, ----, D_7 ← decode IR (12-14),
AR ← IR(0-11) , I ← IR (15)



→ T_0 : AR ← PC
- Place the content of PC onto bus by making bus selection inputs S_2, S_1, S_0 = 010
- Transfer the content of bus to AR. by enabling the load (LD) input in AR

→ T_1 : IR ← M[AR] , PC ← PC+1
- Enable the read i/p of memory.
- Place the content of memory onto bus by making S_2, S_1, S_0 = 111
- Transfer the content of bus to IR by enabling the LD i/p. of IR.
- Increment PC by enabling the INR i/p of PC.

Determine the type of instruction.


Start
SC ← 0
T0
AR ← PC
IR ← M[AR] , PC ← PC+1
T1
Decode opcode in IR (12-14)
AR ← IR (0-11) , I ← IR (15)
T2
(Register or I/0) = 1
D7
=0 (memory-reference) = opcode = 111
(I/0) =1
I
=0 (register)
(indirect) =1
I
=0 (direct)
T3
Execute input-output Instruction
SC ← 0
T3
Execute register-reference instruction
SC ← 0
T3
AR ← M[AR]
T3
Nothing
Execute memory-reference instruction.
SC ← 0
T4

$D'_7 I T_3$ AR ← M[AR]
$D'_7 I' T_3$: Nothing
$D_7 I' T_3$: Executed a register reference instruction.
$D_7 I T_3$: Execute an input-output instruction.

HLT ⇒ Halt

SPA → skip if AC +ve

SNA → skip if AC -ve

SZA → skip if AC = 0

SZE → skip if E = 0

Date: 4/10/18

Page:

Register - reference Instructions

$r = D_7 I' T_3$ → Register reference Instruction.

$B_i = IR(i)$, $i = 0, 1, 2, ---, 11$.

	r:	$SC \leftarrow 0$
CLA	rB_{11} :	$AC \leftarrow 0$
CLE	rB_{10} :	$E \leftarrow 0$
CMA	rB_9 :	$AC \leftarrow AC'$
CME	rB_8 :	$E \leftarrow E'$
CIR	rB_7 :	$AC \leftarrow shr\ AC$; $AC(15) \leftarrow E$, $E \leftarrow AC(0)$
CIL	rB_6 :	$AC \leftarrow shl\ AC$; $AC(0) \leftarrow E$, $E \leftarrow AC(15)$
INC	rB_5 :	$AC \leftarrow AC+1$
SPA	rB_4 :	if $(AC(15)=0)$ then $PC \leftarrow PC+1$
SNA	rB_3 :	if $(AC(15)=1)$ then $PC \leftarrow PC+1$
SZA	rB_2 :	if $(AC=0)$ then $PC \leftarrow PC+1$
SZE	rB_1 :	if $(E=0)$ then $PC \leftarrow PC+1$
HLT	rB_0 :	$S \leftarrow 0$ (S is a start-stop flip-flop)

Memory - reference Instruction.

BUN → Branch unconditionally.

BSA → Branch and save return address.

→ The effective address of the instructions is in AR, and was placed there during timing signal T_2 when $I=0$, or during timing signal T_2 when $I=1$.

→ The execution of MR instruction starts with T_4

LDA → load to AC

STA → store AC.

Date:

Page:

symbol	operation decoder.	symbolic description.
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR]$ $E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC$, $PC \leftarrow AR+1$
ISZ	D_6	$M[AR] \leftarrow M[AR]+1$, if $M[AR]+1=0$ then $PC \leftarrow PC+1$

→ Memory cycle is assumed to be short enough to complete in a CPU cycle.

AND to AC

D_0T_4 : $DR \leftarrow M[AR]$ Read operand

D_0T_5 : $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$ AND with AC

ADD to AC

D_1T_4 : $DR \leftarrow M[AR]$ Read operand

D_1T_5 : $AC \leftarrow AC + DR$, $E \leftarrow C_{out}$ $SC \leftarrow 0$ Add to AC and store carry in E

BUN (Branch unconditionally)

This instruction transfer the program to the instruction specified by the effective address.

D_4T_4 : $PC \leftarrow AR$, $SC \leftarrow 0$

BSA (Branch and save return address)

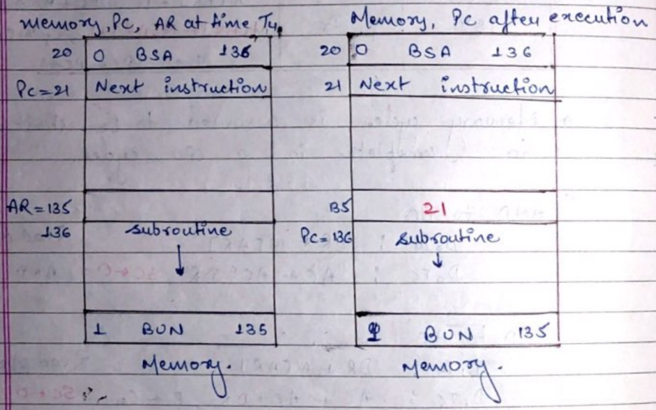
This instruction is useful for branching to a portion of program called subroutine or procedure. When executed, BSA instruction store the address of next instruction in sequence(

Date:
Page:

available in PC) into a memory location specified by the effective address. The effective address+1 is then transferred to PC. to serve as the first instruction in subroutine.

M[AR] ← PC , PC ← AR+1

4/10/18



136 ← PC , PC ← 21

ISZ (increment and skip if zero)

M[AR] ← M[AR]+1 , if M[AR]+1=0 then PC ← PC+1

(numbers (memory content) is signed number, sign bit =1, mag = 2's complement if it's -ve number)

Ques. Explain why each of the following micro-opⁿ can't be executed during a single clock pulse in the system. Specify the

Date: 4/10/18
Page:

sequence of micro-opⁿ that will perform the operation.

1) IR ← M[PC] 2) ~~A~~ AC ← AC+TR
3) DR ← DR+AC

Not possible in single clock pulse. because:

1) AR ← PC.
IR ← M[AR]
PC can't provide address for memory so first transfer to AR. then to IR.

2) DR ← TR
AC ← AC+DR
Add opⁿ must be done with DR.

3) DR ← AC, AC ← DR.
AC ← AC+DR
DR ← AC, AC ← DR.

(chapter - 7)

MICRO-PROGRAMMED Control UNIT

→ The control function that specifies a micro-opⁿ is a binary variable. when it is in 1 binary state the corresponding micro-opⁿ is executed.

→ Control unit initiates the series of sequential steps of micro-operations, during any given time certain micro-opⁿ are to be initiated while others remain idle.

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

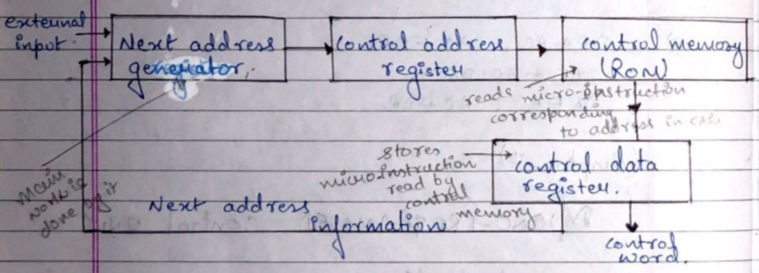
*→ instruction / micro-instr diff = ?

Date:
Page:

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

→ Each word in control memory contains within it a micro-instructions. and a sequence of microinstructions constitute microprogram for system.
Microinstr. specify one or more micro opⁿ for the system

9/10/10 Block diagram of micro-programmed control unit.



→ The control memory is assumed to be in ROM within which all control information is permanently stored.

→ The control memory address reg. (CAR) specifies the address of micro-instruction and control data register holds the micro instruction read from control memory.

→ The micro-instruction contains a control word that specifies one or more micro-opⁿ for data processor. Once these opⁿ are executed, the control must determine the next address. The location of next micro-instruction may be the one next in sequence or it may be located somewhere else in control memory. For this reason, it is necessary to use some bits of present micro-instruction to control the generation of the address of next micro-instruction.

→ The next address generator is sometimes called micro-programmed sequencer as it determines the address sequence i.e. read from control memory. The functions of microprogram sequencer are increment the CAR by 1, loading into CAR and address from memory, transferring an external address or loading an initial address to start the control operation.

→ The control data register holds the present microinstruction while the next address is computed & read from memory. The control data reg. is sometimes called a pipeline register.

10/10/10 Address Sequencing
Microinstructions are stored in control memory in groups with each group specifying a routine.
Each computer instruction has its own micro-programmed routine in control memory to generate the micro-opⁿ. that execute

the instruction.

→ The control must perform the following steps during the execution of single computer instruction.

step (i) An initial address is loaded into CAR when power is turned on in the computer. This address is the address of the first micro-instr. that activates the instruction fetch routine. The fetch routine is sequenced by ~~#~~ incrementing the CAR through the rest of its micro-instruction. At the end of fetch routine the instruction is in the instruction register of the computer.

step (ii) The control memory next go through the routine that determines the effective address of operand. The effective address computation routine in control memory can be reached through a branch micro instruction which is conditioned on the status of the mode bits of the instruction. When the EA computation routine is completed the address of the operand is available in memory address register.

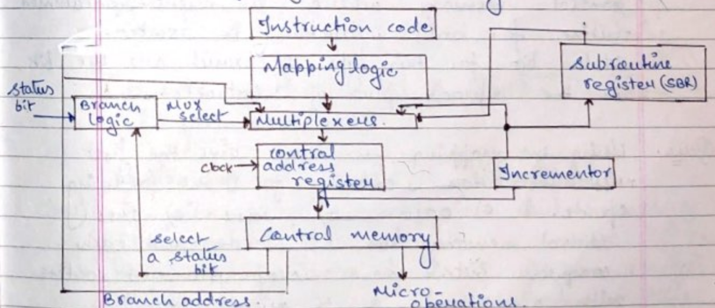
step (iii) The next step is to generate the micro-op^n that execute the instruction fetched from memory. The micro-op^n steps to be generated in processor registers depend upon the opcode part of the instruction. The transformation from instruction code bits to an address in control memory where the routine is located is known as mapping.

Date: 10/10/18

Once the required routine is reached, the micro-instruction that execute the instruction is sequenced by incrementing CAR.

step (iv) When the execution of instruction is completed the control must return to fetch routine. This is accomplished by executing an unconditional branch micro-instruction to the first address of the fetch routine.

* The address sequencing capabilities required in 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 instruction to an address for control memory.
(iv) A facility for subrouting call & return.



Selection of address for control Memory

Date: 11/10/18
Page:

Conditional branching.
The branch logic provides decision making capabilities in the control unit.
The simplest way is to test the specified condition and branch to the indicated address if the condition is mapped otherwise address register is incremented.

An unconditional ~~branch~~ microinstr. can be implemented by loading the branch address from control memory into CAR. This can be accomplished by fixing the value of 1 status bit at the i/p of Multiplexer so, it is always equal to 1.

Mapping of instruction.
A special type of branch exists when a micro-instr. specifies a branch to first word in control memory where a micro-programmed routine for an instruction is located.
$\underbrace{\text{Status bit}}$ for this type of branch are the bits in the opcode part of instruction.

Ques. Using the mapping procedure give the first micro instruction address for the following opcode. a) 0010 b) 1011 c) 1111
Control memory has 128 words and each computer instr. has microprogrammed routine with a capacity of 4 micro instr.

a) 8 to 11 c) 60 to 63
b) 44 to 47

0 0010 00
0 1011 00
32 + 8 + 4 = 44
32 + 16 + 8 + 4 = 60

$0\,\underset{xxxx}{1011}\,\underline{00} = 44$

$\underline{0}\,\underset{x\,x\,x\,x}{0010}\,\underline{00} = 8$

Hybrid = vertical + horizontal.
↳ speed = not too slow not too fast
size = optimal
parallelism depend on horizontal.

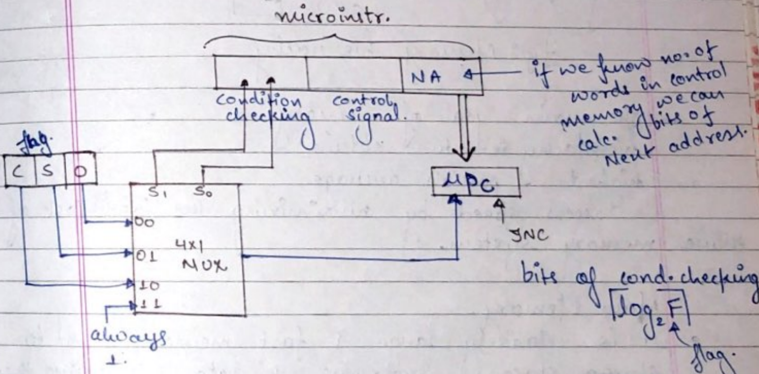
Date: 11/10/18
Page:

Ques. Formulate a mapping procedure that provides 8 consecutive microinstruction for each routine. The opcode has 6 bits and control memory has 2048 words.
$\underline{8}$ → (3) → zero in right

$2048 = 2^{11}$

11 < 6 opcode, 3 right, 2 left

$00\,\underbrace{xxxxxx}_{6\text{ bit opcode}}\,000$



bits of cond. checking $\lceil \log_2 F \rceil$ ← flag.

Concept of horizontal & vertical micro-programming.

	Horizontal		Vertical.
1.	The size of control word is large.	1.	Control word is small.
2.	Perform many control signal simultaneously.		Perform 1 control signal at a time.
3.	Parallelism is more.		There is no Parallelism.
4.	Faster.		Slower.