FEBRUARY 2012

Chapter - 2

February 22 Wednesday

Basic Computer Organization and Design

Instruction code :-

Instruction code is a group of bits that instruct the computer to perform a specific operation. It is divided into parts, each having its own ~~interpretation~~ meaning. The most basic part of an inst. code is its operation part.

Operation code :-

February 23 Thursday

The operation code of an inst. is a group of bits that define such operations as add, subtract, multiply, shift etc. The no. of bits required for the operation code of an inst. depends on the total no. of operations available in the computer.

Stored Program Organization :-

Inst. code format with two parts. First part specifies the operation to be performed and second part specify the address. For a memory unit with 4096 words we need 12 bits to specify an address ie $2^{12} = 4096$. If we store each inst. code in one 16-bit memory word, we have four bits for

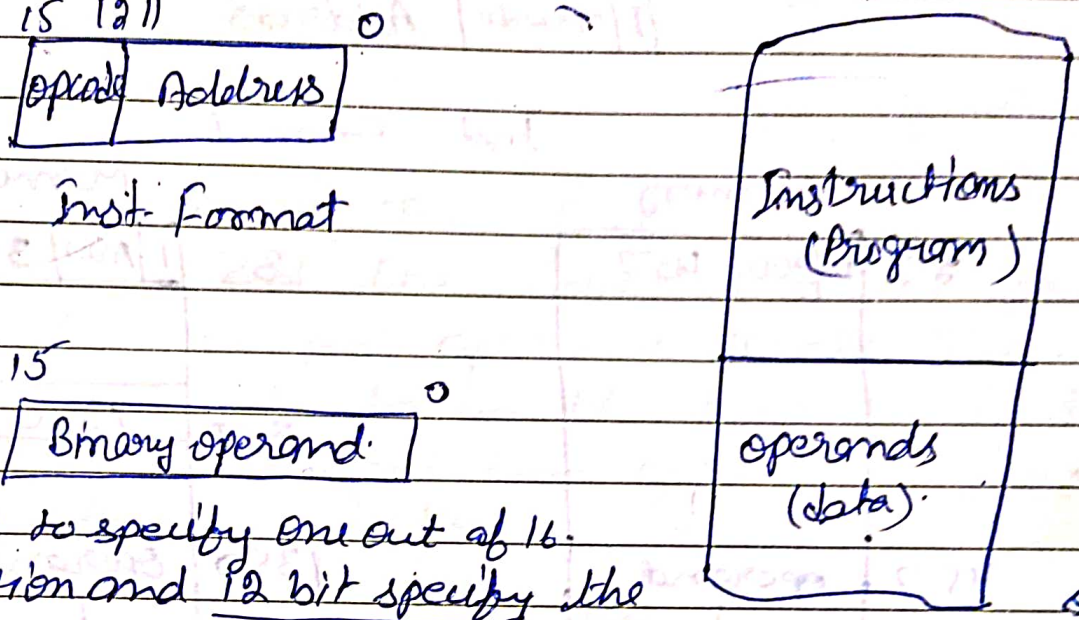
$2^{12} = 4096$ = 4096

address lines = 12

FEBRUARY 2012

Memory: 4096 X 16

February 24 Friday



opcode to specify one out of 16 possible operation and 12 bit specify the address of an operand. The control reads 16-bit inst from program portion of memory. It uses 12-bit address part of inst

specified by the opcode

Processor Reg. AC

February 25 Saturday

to read 16-bit operand from data portion. It then execute the operation.

⇒ When the second part of an inst. code specify an operand, the inst. is said to have an immediate operand.

⇒ When the second part specifies the address of operand, the instruction is said to have direct address.

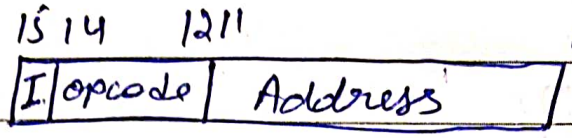
February 26 Sunday

⇒ When the second part specifies the address of memory word in which the address of the operand is found is called indirect address.

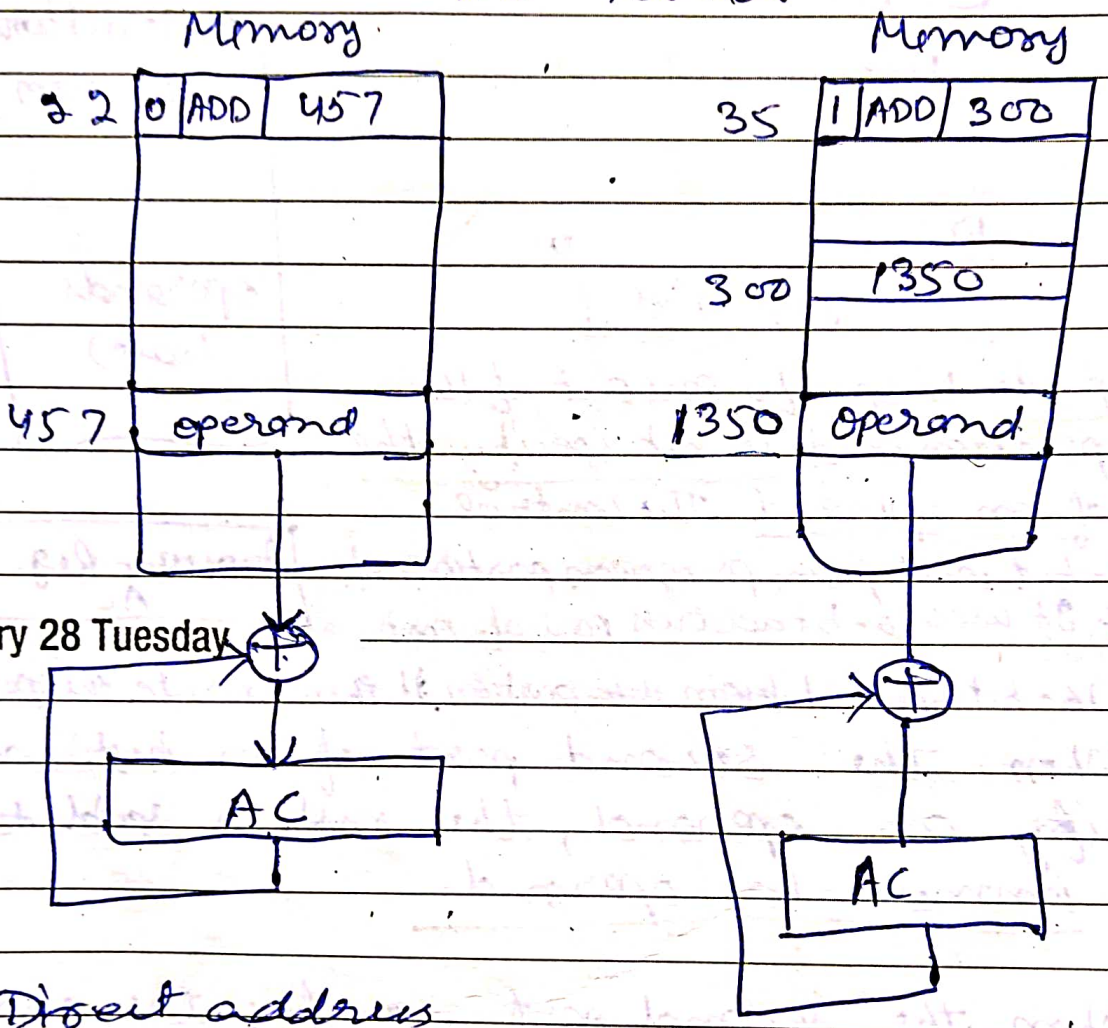
One bit of the inst. code can be used to diff. b/w direct and indirect address.

FEBRUARY 2012

February 27 Monday



Inst Format



February 28 Tuesday

Direct address

Indirect Address

We have a Inst format which consist of a 3 bit opcode, 12 bit address and an address mode bit designated by I. The mode bit is 0 for a

FEBRUARY 2012

February 29 Wednesday direct address and 1 for an indirect address.

The indirect address reference is needs two references to memory to fetch an operand. The first reference is needed to read the address of the operand and second is for operand itself. The effective address to be the address of the operand in a computation type inst. or the target address in a branch-type inst.

effective address is 457 in direct addressing mode

Notes effective address is 1350 in indirect addressing mode

Computer Registers:-

Registers are used in control unit for storing the instcode after it is read from memory

List of Registers:-

Reg. symbol	No. of bits	Reg name	Function
DR	16	Data Reg.	hold mm operand
AR	12	Address Reg.	Addr for memory
AC	16	Acc	Processor Reg.
IR	16	Inst. Reg.	hold inst. code
PC	12	Program counter	Add. of inst

MARCH 2012

0000-0
0001-1
0010-2
0011-3
0100-4
0101-5
0110-6
0111-7

1000-8
1001-9
1010-A
1011-B
1100-C
1101-D
1110-E
1111-F

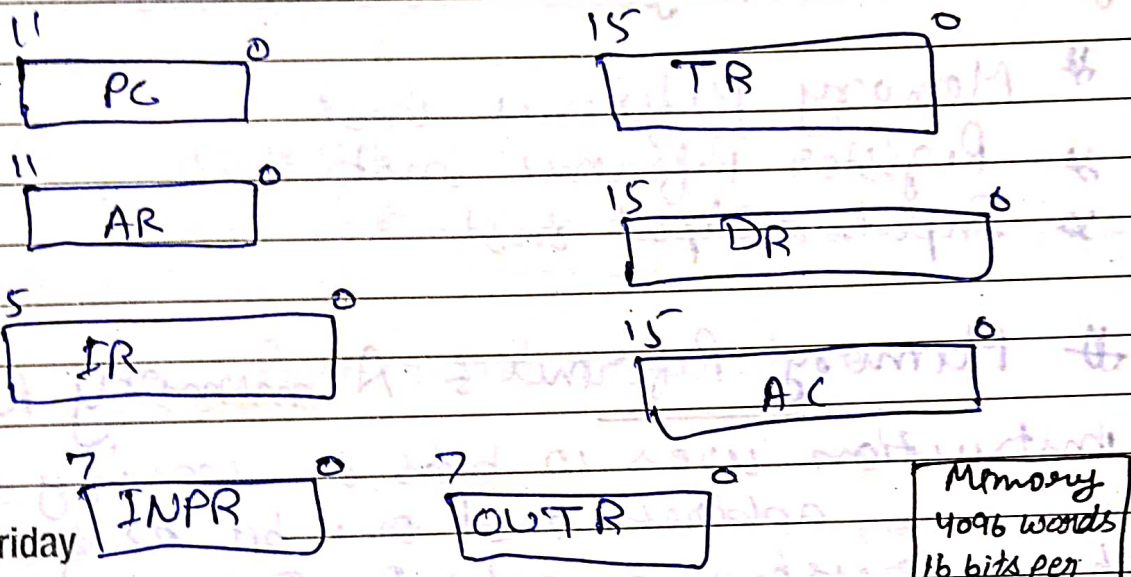
(5)

Monthly Planner

MARCH 2012

March 01 Thursday

TR	16	Temporary Reg. Holds temp. data
INPR	8	Input Reg. hold input character
OUTR	8	output Reg. hold output character



March 02 Friday

Basic computer registers

* Data Reg :- DR holds the operand read from memory.
* Acc Reg :- AC reg. is a processing register.
* IR :- The instruction read from memory is placed in the instruction register.
* TR :- The temporary register is used for holding temporary data during the processing.
* AR :- AR is used to hold the address of memory.
* PC :- PC has 12 bits and it holds the address of the nxt instruction to be executed.
* Input Reg :- INPR receives an 8-bit character from an input device.

Output Reg – 8 Bit register – send 8 bit characters to an output device

ADD A, 25

MARCH 2012

March 03 Saturday

Computer Instructions:

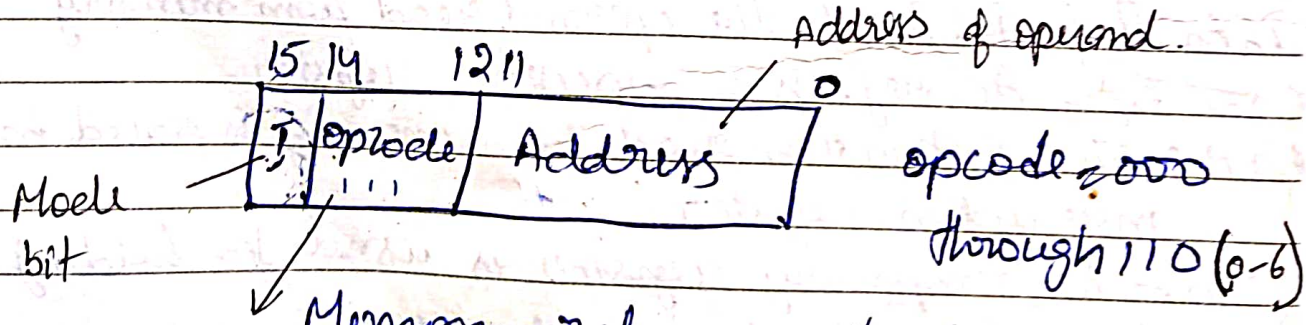
* **Instruction format:** The basic computer has three instruction code format. Each format has 16 bits.

* Memory Reference inst
* Register Reference inst.
* Input-output inst.

* **Memory Reference:** A memory Reference instruction uses 12 bits to specify an

March 04 Sunday

address and one bit to specify the addressing mode I. I is equal to 0 for direct addressing and 1 for indirect address mode.



opcode

000 – 0
001 – 1
010 – 2
110 – 6

0 – direct Addressing
1 – Indirect Addressing

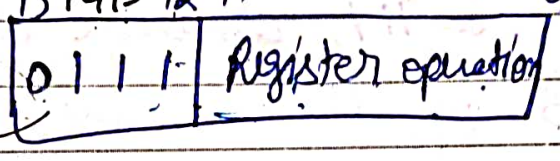
MOV AX, BX (8) RR RR

I/O 6
111

MARCH 2012

March 05 Monday

* Register-Reference Inst:- Register-reference inst. specifies an operation on AC register. An operand from memory is not needed therefore the 12 bits are used to specify the operation (Register)



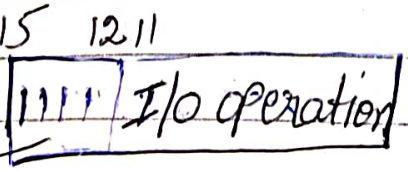
(opcode = 111, I = 0)

Always equal to 7

Register-reference inst.

March 06 Tuesday

* Input-output Inst:- An input-output inst. does not need a reference to a memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of input-output operations.



(opcode = 111, I = 1)

Always equal to F

I/O instruction.

MARCH 2012

March 07 Wednesday The type of inst is recognized by the computer control from the four bits in positions 12 through 15 of the inst. If the three opcode bits in positions 12 thro. 14 are not equal to 111, the instruction is a memory Reference type and the bit in position 15 is taken as the addressing mode I. If the 3-bit opcode is equal to 111, control inspects the bit in position 15. If it is 0, the inst. is register reference type. If the bit is 1, the instruction is an input-output type. The addressing mode bit I is not used as a mode bit when the

March 08 Thursday operation code is equal to 111.

Memory Reference Inst / Basic computer instructions:

Symbol	Hexadecimal code I=0	I=1	Description
AND	0XXX	8XXX	LDA
ADD	1XXX	9XXX	STA
			BUN
			BSA
			ISZ

1XXX: binary equivalent of opcode 1; XXX: Address bits

when I=0 the bits from 12-14

0 — 6

when I=1 the bits from 12-14

8-E

MARCH 2012

Register-Reference Inst:-

March 09 Friday

CLA	7800	— Clear AC
CLE	7400	— Clear E
CMA	7200	— complement AC
CME	7100	— complement E.

0 | 111 | 001

CIR, CIL, INC
SPA - Skip nxt inst. if AC +ve
SNA
SZA, SZE, HLT

Input/output Reference:-

INP F800 — Input character to AC.

March 10 Saturday

1111

F

OUT F400 — Output character from AC.

ION, IOF, SKI, SKO.

Instruction Set:-

1. Arithmetic, logical and shift instructions.
2. Instructions for moving information to and from memory and processor registers.

March 11 Sunday

3. Program control instructions.
4. Input and output instructions.

MARCH 2012

March 12 Monday

Common Bus System :

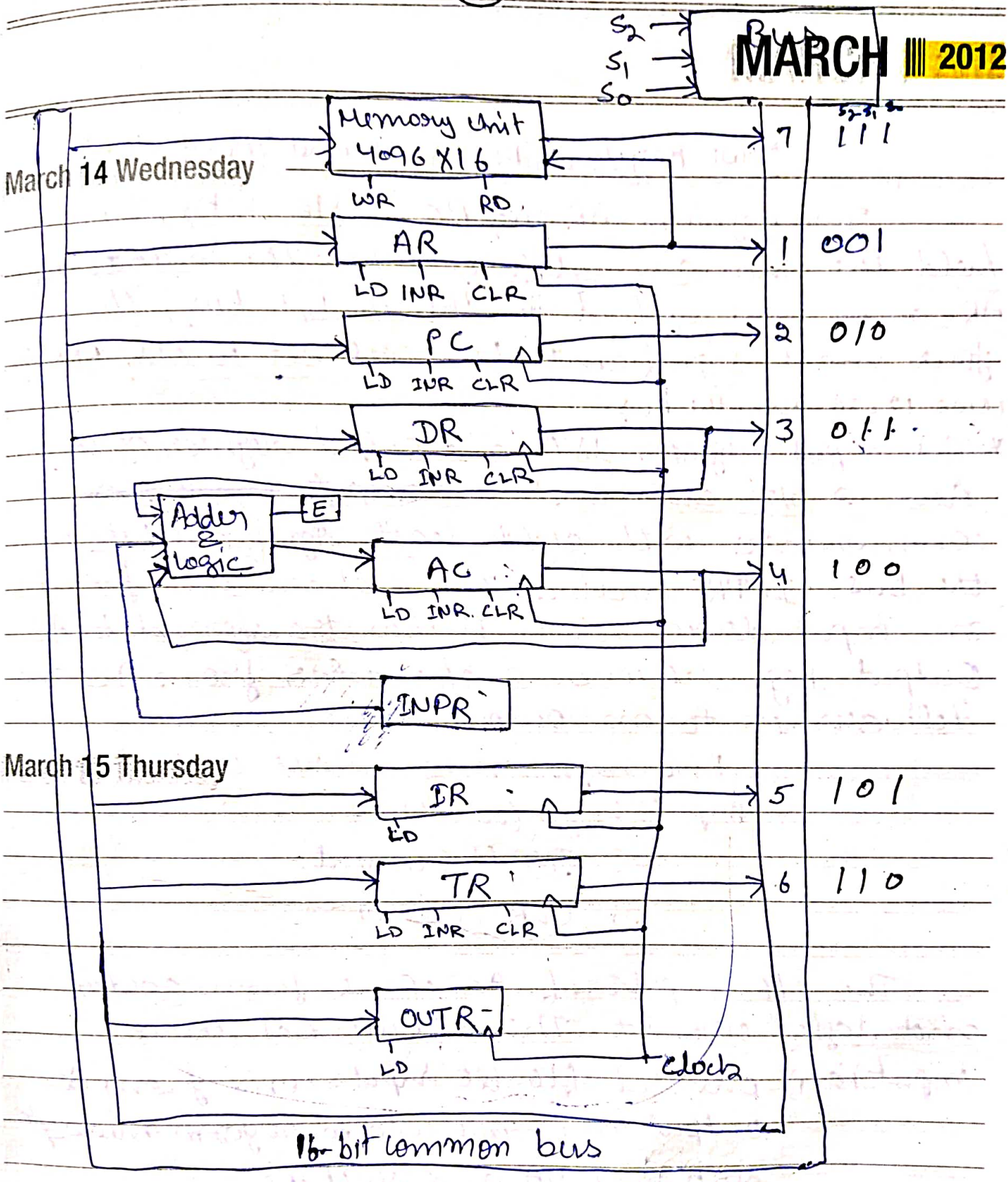
The basic computer has eight registers, a memory unit and a control unit. Paths must be provided to transfer information from one register to another and between memory and registers. The no. of wires will be excessive if connections are made b/w the outputs of each register and inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus. The connection of the registers and memory

March 13 Tuesday

of the basic computer to a common bus is shown in fig.

The output of seven registers and memory are connected to common bus. The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S_2, S_1, S_0.

e.g :- The number along the output of DR is 3. The 16-bit outputs of DR are placed on the bus lines when $S_2S_1S_0 = 011$, this is the binary value of decimal 3. The lines from common bus are connected to the inputs of each register and memory. The particular register whose load LD (load) input is enabled



receives the data from the bus. The memory places its 16-bit output onto the bus when the read input is activated and $S_2S_1S_0 = 111$.

MARCH 2012

March 16 Friday Four Registers DR, AC, IR and TR — 16 bits.

Two registers AR and PC — 12 bits, they hold the memory address. When the contents of AR or PC are applied to the 16-bit bus, the four most significant bits are set to 0's. AR&PC receive 12 LSB from the bus.

* The input register INPR and output register OUTR have 8 bits each and ~~the output register~~ communicate with eight least significant bits in the bus. INPR receives a character from an input device which is then transferred to AC. Output Reg. receives a character from AC and delivers it to an output device

March 17 Saturday Five registers have three control signals

inputs: LD (load)
: INR (increment)
; CLR (clear)

The 16 inputs of AC come from adder and logic circuit. This circuit has three input ie 1) one set of 16-bit inputs come from AC output. They are used to perform microop^n such as CMA and shift AC.

March 18 Sunday

2) Another 16-bit inputs come from the DR. The inputs from DR and AC are used for arithmetic and logic micro-op^n such as add DR to AC or AND DR to AC.

MARCH 2012

March 19 Monday The result is transfer to AC and end carry moves to flip-flop E.

3) A third set of 8-bit inputs come from input register INPR.

Eg:

DR ← AC

This can be done by placing the content of AC on the bus (with $S_2S_1S_0 = 100$), enabling the LD (load) input of DR.

Timing and Control:-

March 20 Tuesday The clock pluse are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit. The clock pulses do not change the state of a register unless the register is ~~control~~ enabled by a control signal.

There are two types of control org:-

① Hardwired control.

② Microprogrammed control.

Control unit:- It consists of two decoders, a sequence counter and a number of control logic gates. An instruction read from memory is placed in the

MARCH 2012

Control unit: It direct the main operations by sending control signals to data path. It flows b/w CPU, I/O, memory subsystem etc.

March 21 Wednesday

in IR which is divided into three parts: the bit I, operation code and bits 0 thro 11. The operation code in bits 12 through 14 are decoded with a 3 x 8 decoder. The eight outputs of the decoder are designated by symbols D_0 through D_7. The 4 bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}.

The sequence counter SC can be incremented or cleared synchronously. Once a counter is cleared to, causing the next active timing signal to be T_0.

March 22 Thursday

Decoder: A decoder is a circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs

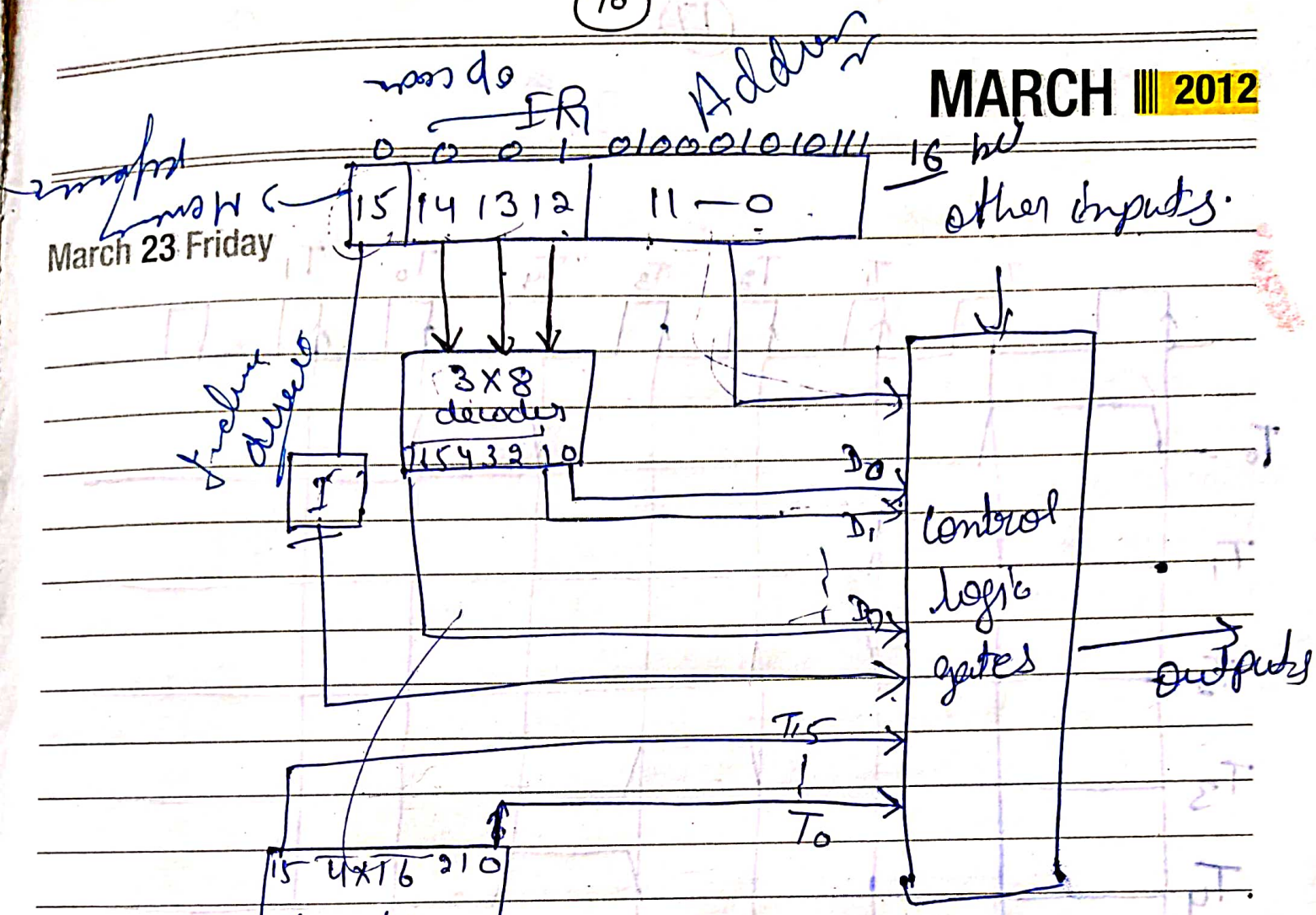
e.g.: Consider SC is incremented to provide timing signals T_0, T_1, T_2, T_3 and T_4 in sequence. At T_4, SC is cleared to 0 if decoder output D_3 is active.

$$D_3 T_4 : SC \leftarrow 0$$

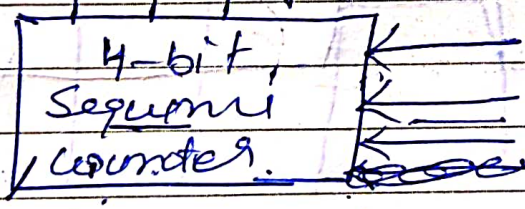
Control signal C_0 & C_1 to make the registers selected in the read mode. Control signals C_2 & C_3 enabled to make the data path enabled so that the content of R1 & R2 available to the data path. ALU enabled by C_4 control signal for addition operation.

MARCH 2012

March 23 Friday



March 24 Saturday



SC = 0

1

2

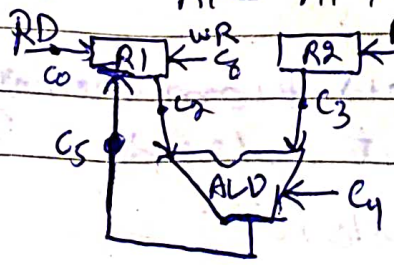
3

Control unit of basic computer

March 25 Sunday

ADD R1, R2

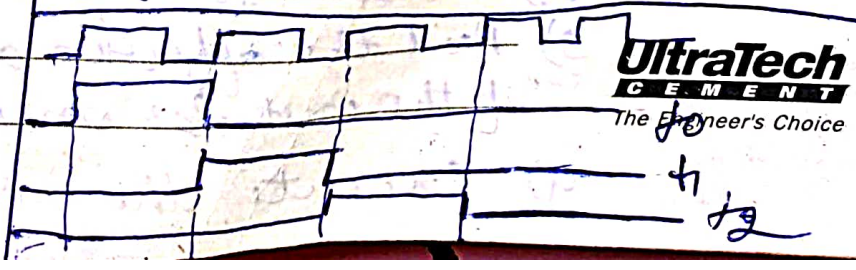
R1 ← R1 + R2



0 0 0 — D_0 — AND

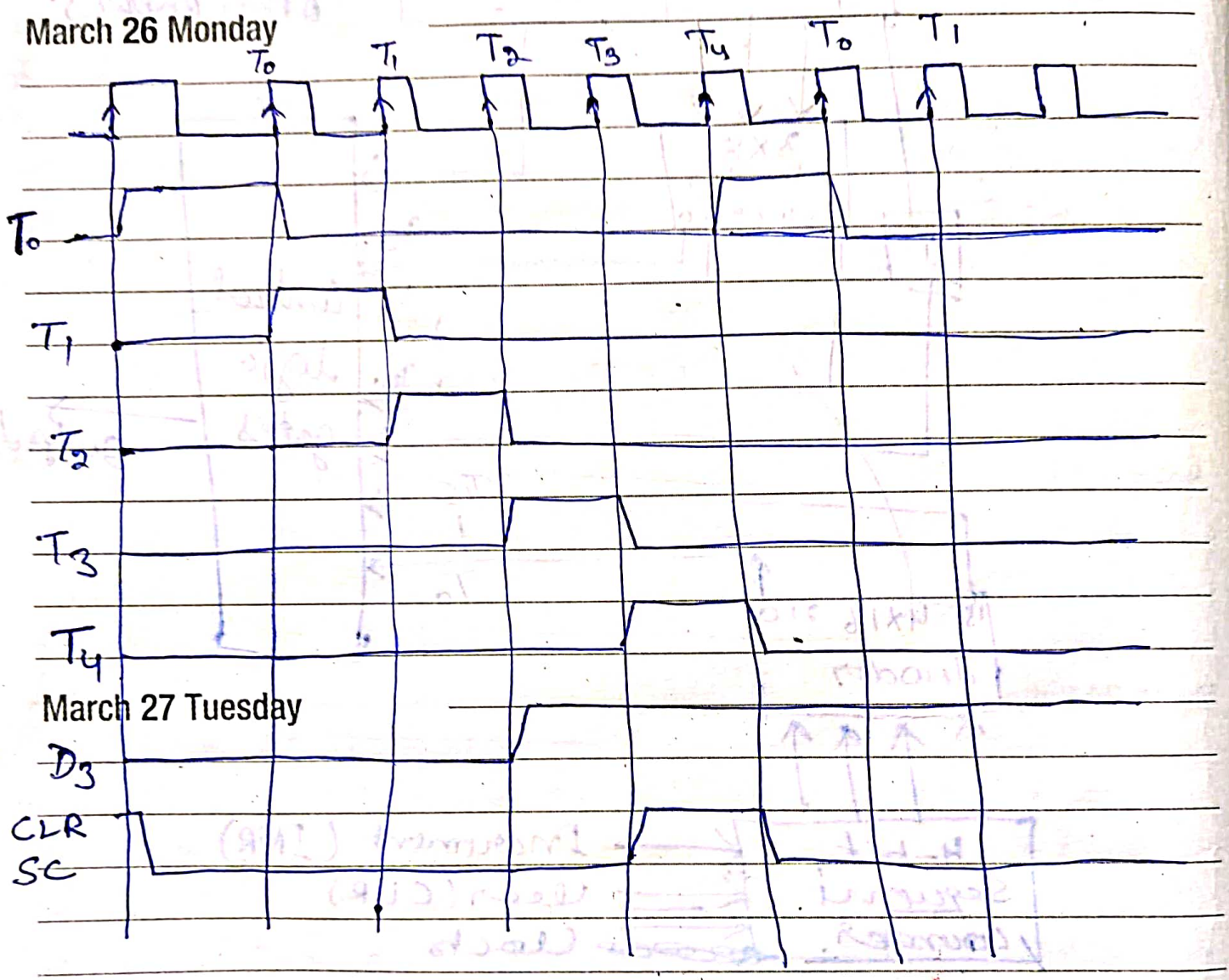
0 0 1 — D_1 — ADD

0 1 0 — D_2 — SUB



MARCH 2012

March 26 Monday



Instruction Cycle :-

A program in memory consist of a sequence of instructions. In basic computer each instruction cycle consist of the following phases:

1) Fetch an instruction from memory
2) Decode the inst
3) Read the effective address from memory if the inst has an indirect address.
4) Execute the instruction.

March 28 Wednesday upon the completion of step 4, the control goes back to step 1 to fetch, decode and execute the next instruction.

Fetch and decode:

Initially, the program counter PC is loaded with the address of the first inst in the program. The SC is cleared to 0, providing a decoded timing signal T_0. The micro-operations for the fetch and decode phases can be specified by the following register transfer statements.

T_0 : $AR \leftarrow PC$

March 29 Thursday T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$

T_2 : $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12-14)$,

$AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$

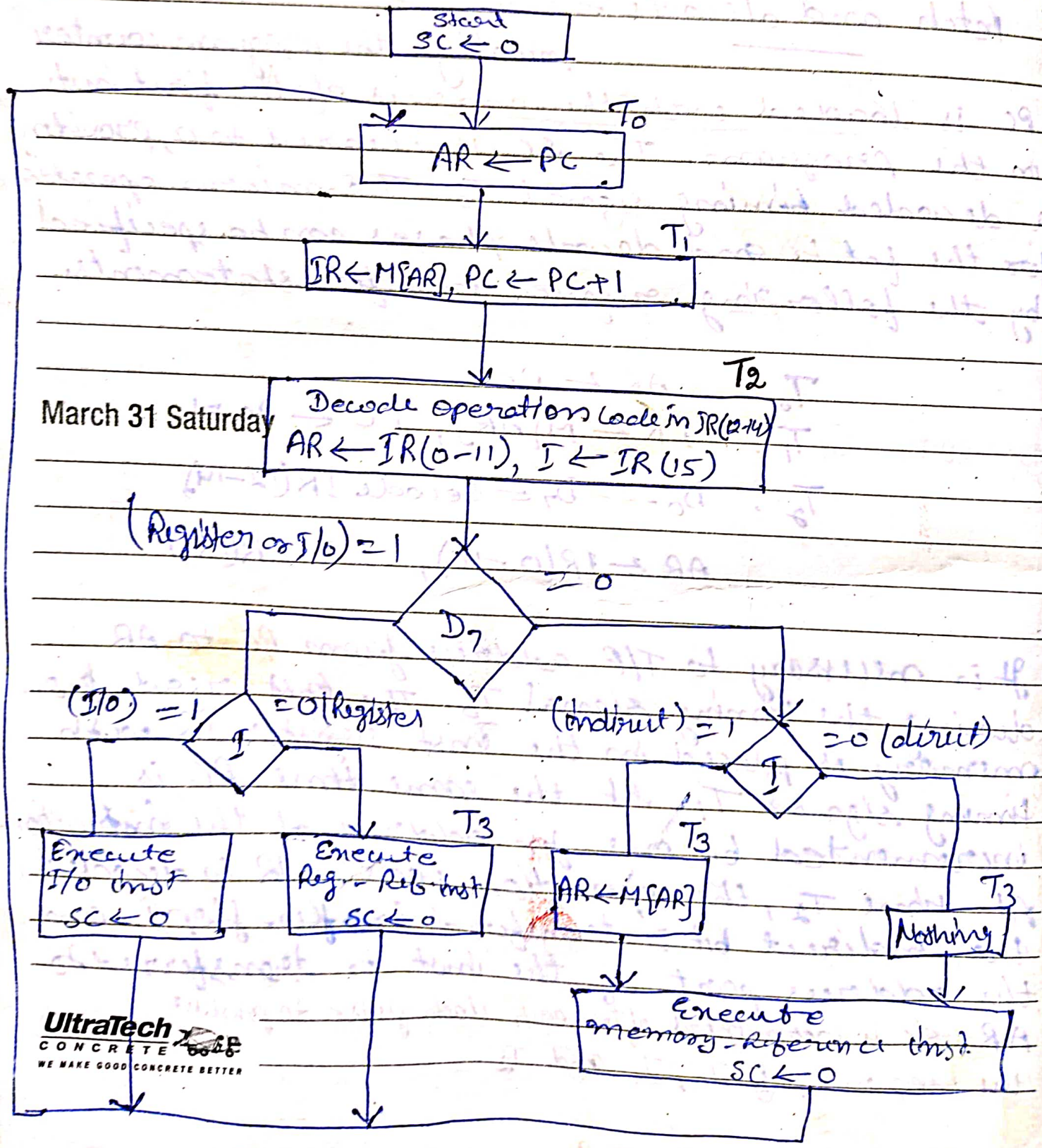
It is necessary to T/P address from PC to AR during the timing signal T_0. The inst read from memory is placed in the inst. register IR with timing signal T_1. At the same time PC is incremented by one for address of the next instruction. At time T_2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the inst is transferred to AR. SC is incremented after each clock pulse to produce the sequence of T_0, T_1 and T_2.

MARCH 2012

March 30 Friday

Determine the type of Instruction :-

During T_3, the control unit determines the type of instruction that was read from memory.



March 31 Saturday

Notes

$D_7' I T_3$: $AR \leftarrow M[AR]$

$D_7' I' T_3$: Nothing

$D_7 I' T_3$: Execute a register-reference inst.

$D_7 I T_3$: Execute an input-output inst.

Register-Reference Instructions:

Register-reference instructions are recognized by the control when $D_7 = 1$ and $I = 0$. These inst. use bits 0 to 11 of the inst. code to specify one of 12 instructions. These inst. are executed with clock transition associated with timing variable T_3.

$D_7 I' T_3 = r$ (common to all register-ref. instructions)

$IR(i) = B_i$ (bit in IR(0-11) that specify the operation)

CLA	$r B_{11}$:	$AC \leftarrow 0$	clear AC
CLE	$r B_{10}$:	$E \leftarrow 0$	clear E
CMA	$r B_9$:	$AC \leftarrow \overline{AC}$	complement AC
			etc.

e.g:

B_{11} D_7 I'

CLA – 7800 — 0111 1000 0000 0000

CLE – 7400

CMA – 7200

APRIL 2012

April 01 Sunday

Memory-Reference Instructions:-

Effective Address:-

List of Memory-Reference Instrs:-

Symbol	operation Decoder	
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow Cout$
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 $PC \leftarrow PC+1$

April 02 Monday

(1) AND to AC:- This perform AND operation on pairs of bits in AC and memory word specified by the effective address. The result is transfered to AC.

$D_0 T_4$: $DR \leftarrow M[AR]$

$D_0 T_5$: $AC \leftarrow AC \wedge DR, SC \leftarrow 0$

(2) ADD to AC:- This add the content of memory word specified by the effective address to the value of AC. The sum is transferred to AC and the output carry Cout is transferred to E flip-flop.

$D_1 T_4$: $DR \leftarrow M[AR]$

$D_1 T_5$: $AC \leftarrow AC + DR, E \leftarrow Cout, SC \leftarrow 0$

APRIL 2012

April 03 Tuesday ③ LDA: Load to AC:-

This transfers the memory word specified by the effective address to AC.

D_2T_4: DR ← M[AR]

D_2T_5: AC ← DR, SC ← 0

④ STA: Store AC:

This inst. stores the content of AC into memory specified by the effective address.

D_3T_4: M[AR] ← AC, SC ← 0

April 04 Wednesday ⑤ BUN: Branch Unconditionally:

This inst. transfers the program to the instruction specified by the effective address. PC holds the address of the inst. to be read from memory in the next inst. cycle. PC is incremented at the time T_1 to prepare it for the address of the next inst. in the sequence. The BUN inst. allows the programmer to specify an inst. out of sequence and we say that the program branches or jumps unconditionally.

D_4T_4: PC ← AR, SC ← 0

The next inst is fetched and executed from the memory address given by new value in PC.

APRIL 2012

April 05 Thursday

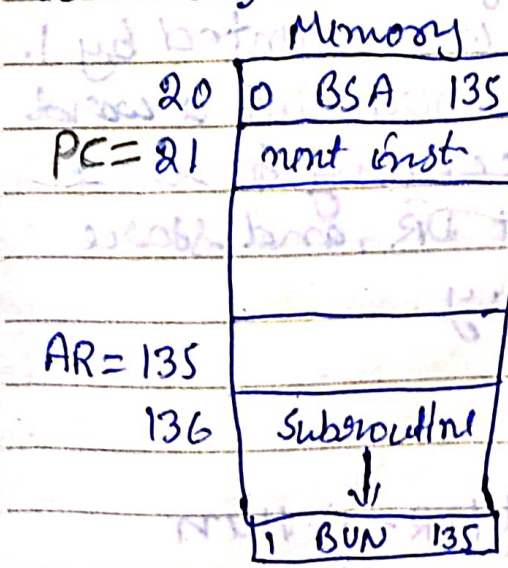
BSA: Branch and Save Return Address:-

This inst is useful for branching to a positional program called a subroutine or procedure. When executed, the BSA inst. stores the address of the next instruction in sequence (which is available in PC) into a memory location specified by the effective address. The effective address plus one is then T/F to PC. to serve as the address of first instruction in the subroutine.

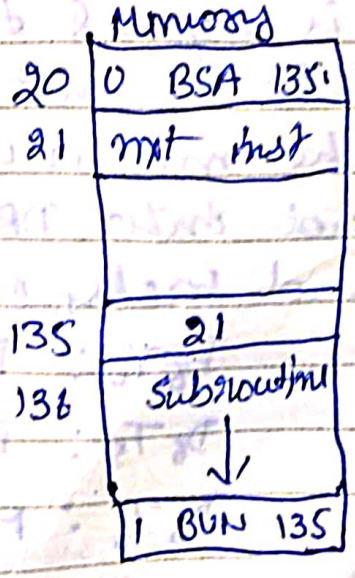
$$M[AR] \leftarrow PC, \quad PC \leftarrow AR+1.$$

April 06 Friday

e.g:- Suppose BSA inst. is in memory address 20. The I=0 and address part of the inst. has binary equivalent of 135. After the fetch and decode phase, PC contains 21, which is the address of the next inst. in the program (referred to as return address). AR holds the effective address 135.



Memory, PC and AR at time T_4



Memory & PC after execution.

APRIL 2012

April 07 Saturday $M[135] \leftarrow 21$, $PC \leftarrow 135+1 = 136$.

The return address 21 is stored in memory location 135 and control continues with the subroutine program starting from address 136. The return to the original program is accomplished by means of an indirect BUN inst placed at the end of the subroutine. When BUN inst. is executed, the effective address 21 is transferred to PC

$D_5 T_4$: $M[AR] \leftarrow PC$, $AR \leftarrow AR+1$

$D_5 T_5$: $PC \leftarrow AR$, $SC \leftarrow 0$

April 08 Sunday

(6) ISZ: Increment and Skip if Zero:-

This inst. increments the word specified by the effective address and if the increment value is equal to 0, PC is incremented by 1. It is not possible to increment a word inside the memory, it is necessary to read the word into DR, increment DR, and store the word back into memory.

$D_6 T_4$: $DR \leftarrow M[AR]$

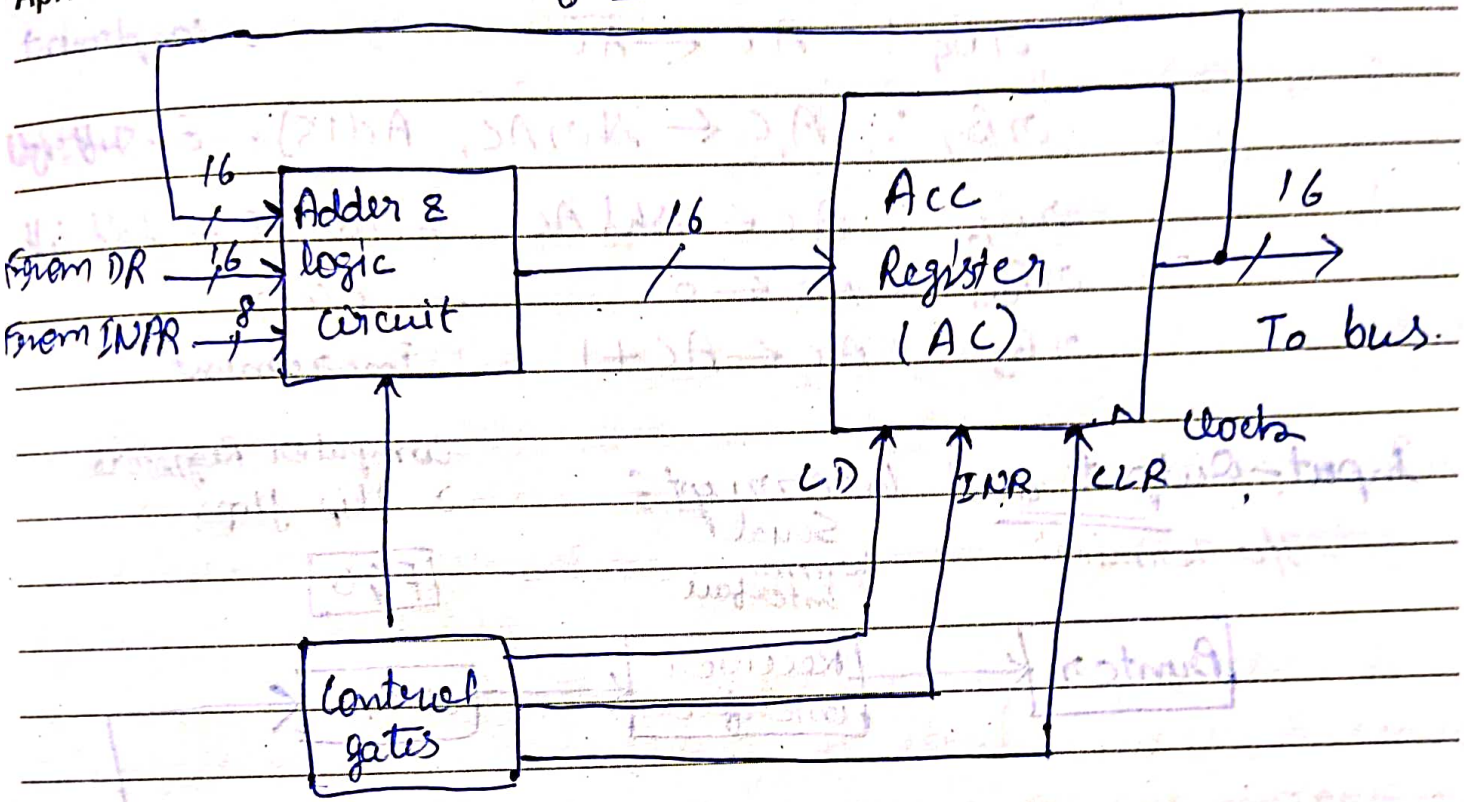
$D_6 T_5$: $DR \leftarrow DR+1$

$D_6 T_6$: $M[AR] \leftarrow DR$, if $(DR=0)$ then $(PC \leftarrow PC+1)$, $SC \leftarrow 0$

APRIL 2012

April 09 Monday

Design of Accumulator logic:-



April 10 Tuesday

Circuit associated with AC.

One set of 16 inputs comes from the output of AC. Another set of 16 inputs comes from the data register DR. A third set of eight inputs comes from the input register INPR. In addition, it is necessary to include logic gates for controlling the LD, INR and CLR in register and for controlling the operation of the adder and logic circuit.

$D_0 T_5$: $AC \leftarrow AC \wedge DR$

$D_1 T_5$: $AC \leftarrow AC + DR$

$D_2 T_5$: $AC \leftarrow DR$

APRIL 2012

April 11 Wednesday

pB_{11} : $AC(0-7) \leftarrow INPR$ — I/P from INPR

rB_9 : $AC \leftarrow \overline{AC}$ — complement

rB_7 : $AC \leftarrow shr\,AC,\ AC(15) \leftarrow E$ — shift right

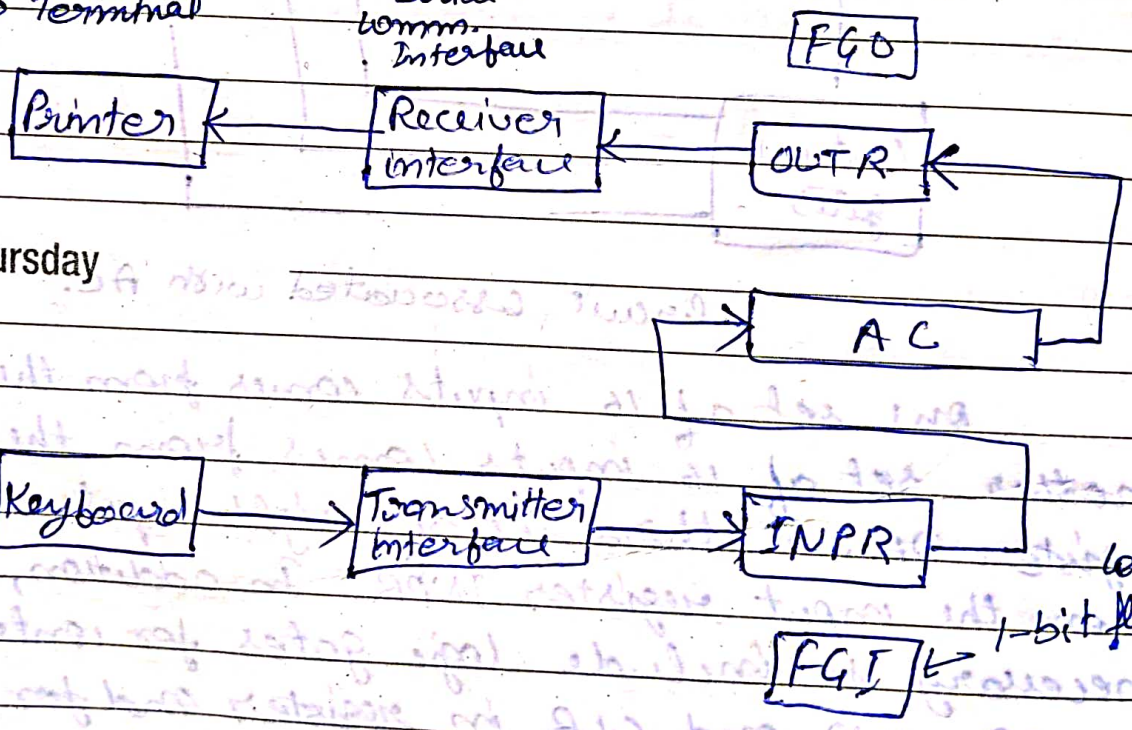
rB_6 : $AC \leftarrow shl\,AC,\ AC(0) \leftarrow E$ — shift left

rB_{11} : $AC \leftarrow 0$ — clear

rB_5 : $AC \leftarrow AC + 1$ — Increment

Input-Output and Interrupt :-

Computer Registers & flip-flops.



April 12 Thursday

The terminal sends and receives serial information. The serial information from the keyboard is shifted into the input register INPR. The serial information for the Printer is stored in output register OUTR. These two registers communicate with a communication interface serially and with AC in parallel.

April 13 Friday The transmitter interface receives serial information from keyboard and transmits it to INPR. The receiver interface receives information from OUTR and sends it to the printer serially.

The input register INPR consist of eight bits and hold an alphanumeric input information. The 1-bit input flag FGI is a control flip-flop. The flag bit is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer.

Initially, the input flag FGI is cleared to 0. When a key is struck in the keyboard

April 14 Saturday an 8-bit alphanumeric code is shifted into INPR and the input flag FGI is set to 1. As the flag is set, the information in the INPR cannot be changed by striking another key. The computer checks the flag bit, if it is 1, the information from INPR is t/f into AC and FGI is cleared to 0. Once the flag is cleared, new information can be shifted into INPR.

The output register OUTR works similar but the direction of information

April 15 Sunday flow is reversed. Initially, the output flag FGO is set to 1. The computer checks the flag bit, if it is 1, the information from AC is transferred in parallel to OUTR and FGO is cleared to 0.

APRIL 2012

April 16 Monday The output device accept the information and when the operation is completed, it sets FGO to 1. The computer does not load new character into OUTR when FGO is 0 because this condition indicates that the output device is in process of printing the character.

Input-Output Instruction;

1) INP pB_{11} AC(0-7) ← INPR, FGI ← 0 Input character.

2) OUT pB_{10} OUTR ← AC(0-7), FGO ← 0 output character.

April 17 Tuesday 3) SKI pB_9: if (FGI=1) then PC ← PC+1 skip on input flag.

4) SKO pB_8: if(FGO=1) then (PC ← PC+1) skip on output flag.

5) ION pB_7: IEN ← 1 Interrupt enable on

6) IOF pB_6: IEN ← 0 Interrupt enable off.

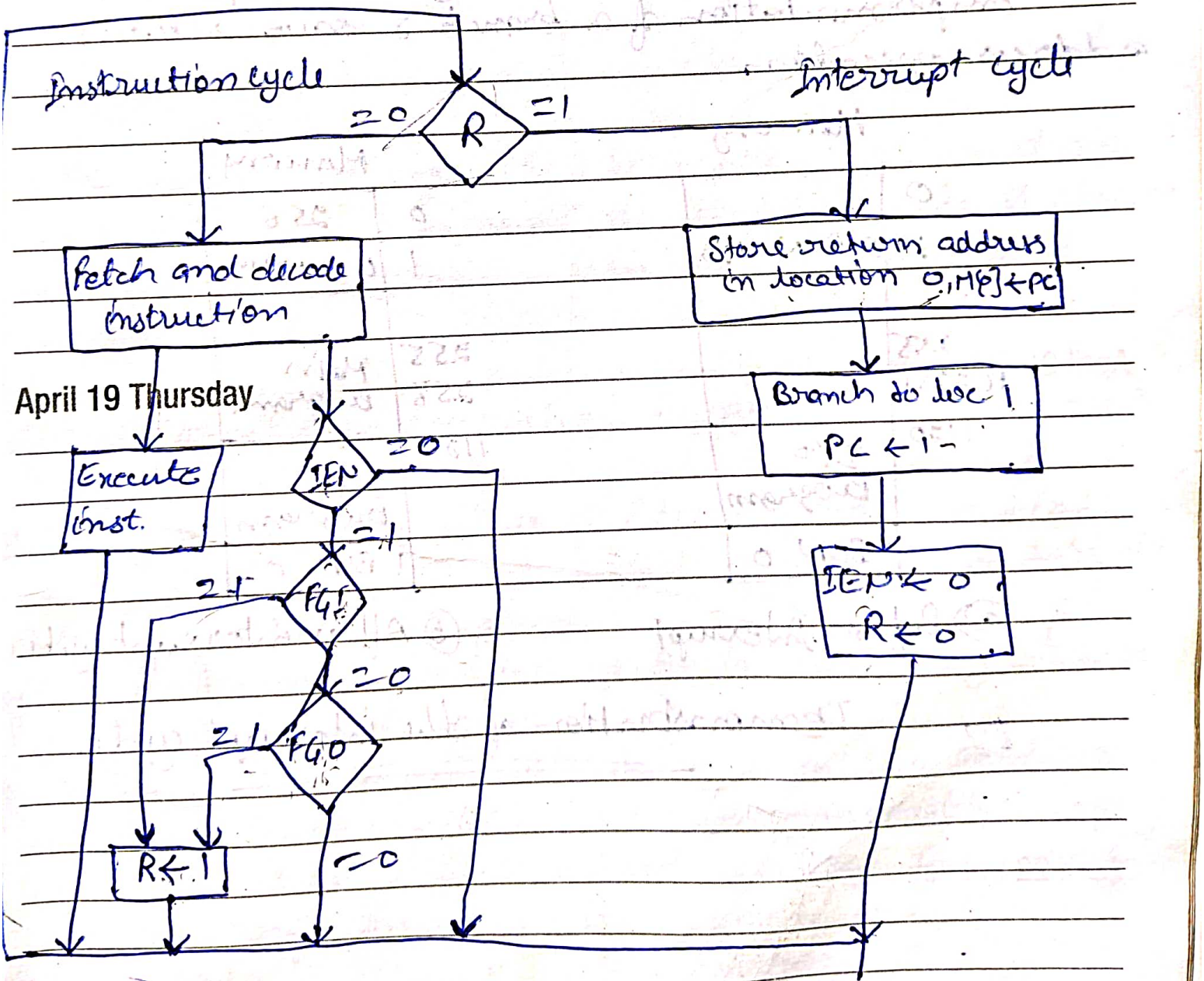
$p = D_7 I T_3$

IR(i) = Bi [bit in IR(6-11) that specifies the instruction].

April 18 Wednesday

Program Interrupt:

The interrupt enable flip-flop IEN can be set and cleared with two instructions. When IEN is cleared to 0 (with IOF inst), the flag cannot interrupt the computer. When IEN is set to 1 (with ION inst), the computer can be interrupted

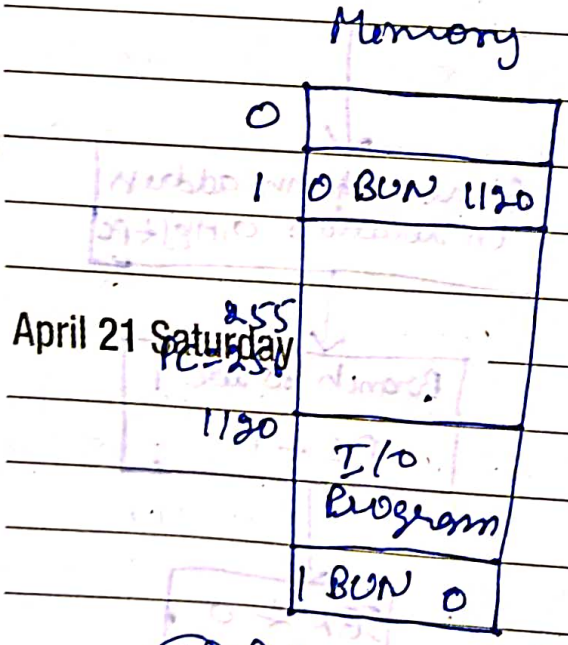


April 19 Thursday

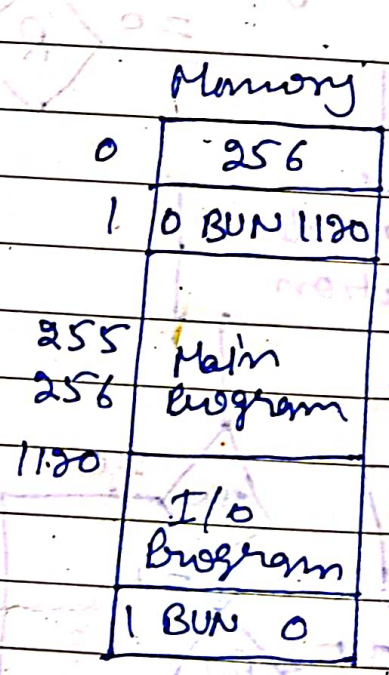
APRIL 2012

April 20 Friday. An Interrupt flip-flop R is included in the computer. When R=0, the computer goes through an instruction cycle. During the execute phase of the instruction phase of the instruction

Interrupt Cycle:- The interrupt cycle is a H/w implementation of a branch & save return address operation.



(a) Before Interrupt



(b) After interrupt cycle

Demonstration of the interrupt cycle.

April 21 Saturday

April 22 Sunday

APRIL 2012

April 23 Monday Control Org:

1) Hardwired
2) Microprogram

Diff. b/w Hard-wired & Microprogram

Hardwired	Microprogrammed.
1) In Hardwired org., the control logic is implimented with gates, flip-flop, decoders etc.	1. In micro-programmed org. the control info is stored in control memory.
April 24 Tuesday	
2) Implimentation approach sequential circuit	2. Programming approach.
3) Not flexible, difficult to modify for new inst.	3. Flexible, new machine inst. can easily be added.
~~4) complicated design proces~~	
4) It can be optimized to perform fast mode of opn	4. It is relatively slower.
5) In Hardwired control, if design has to be modified or change, it requires changes in the wiring among various components	5) Any change or modification can be obtained by changing or updating the micro. program in control memory.
6) RISC microprocessor	6) CISC microprocessor.

APRIL 2012

April 25 Wednesday

7) Example: CPU with hardwired logic control are intel 8085, motorola 6802	7) e.g. CPU with micro programmed control unit are intel 8080, motorola 68000.
8) Control memory is absent i.e combinational circuit is used to generate control signals.	8) Control memory is used to generate control signals
9) Small inst. set size	9) Large inst. set-size
10) Cost of implementation is more	10) Cost of implementation is cheaper.
April 26 Thursday	
11) Very difficult to detect fault	11) Faults can be detected in control unit.

APRIL 2012

April 27 Friday

Program Interrupt :- ① Programmed control Transfer ~~Processor watch the flags~~ The computer keeps checking the flag bit, and when it finds it set, it initiates an information transfer. The computer is wasting time while checking the flag instead of doing some other useful processing tasks.

An alternative to programmed controlled is to let the external device inform the computer when it is ready for the transfer. In the meantime the computer can be busy with other tasks. This type of transfer uses the interrupt facility. While the computer is running a program, it

April 28 Saturday

does not check the ~~interrupt~~ flags. When flag is set, the computer is interrupted from proceeding with current programs and is informed of the fact that a flag has been set. The computer deviates from what it is doing to take care of I/O transfer. It then returns to the current program to continue what it was doing before the interrupt.

April 29 Sunday

APRIL 2012

April 30 Monday

Control Org:-

① Hardwired CU:-

⇒ The control logic is implemented with gates, flip flops, decoders and other digital circuit

⇒ It can be optimized to produce a fast mode of operation

⇒ It requires changes in the wiring among the various components if the design has to be modified or changed.

Microprogrammed CU:-

Note ⇒ The control information is stored in a control memory.

⇒ The control memory is programmed to initiate the required sequence of microoperation

⇒ Any required changes or modifications can be done by updating the microprogram in control memory.