⇒ Opcode & Operands are both types of instrucⁿs. that are used in a microprocessor.

- An Opcode is a command that tells the microprocessor what Operⁿ. to perform, such as adding two nos. or Jumping to a specific locⁿ. in memory.

→ An Operand is the data that the Opcode operates on, such as the no. to be added or the memory locⁿ to Jump to.

- Together an opcode & its operands make up a complete instrucⁿ. that the microprocessor can execute.

Unit - 2

PAGE NO.
DATE : / /

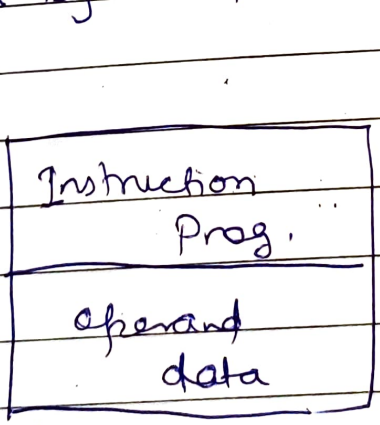
Instruction Codes :- It is a group of bits (01 01 ....) that tells to the ~~processor~~ Computer to perform some specific task.

It can be organized into 2 Parts :-

- opcode
- Address

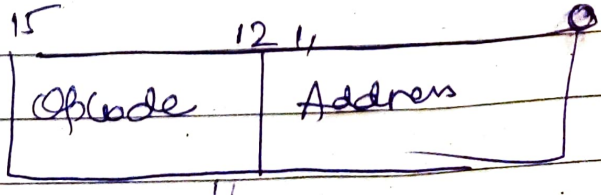
In the memory, the instruction will be present

From the memory, we are getting the instruc" Code & we are going to be converted into 2 groups of bits. & this group of bits will be read by the processor.



Size of memory of
16 bits
4096 × 16
2^{12}
2^4
opcode
12 bits
addr Parts

this instruc" Code can be divided into (16 bits).



Instruction format

PAGE NO. :
DATE : / /

It is of 3 types:-

1) Memory Reference Instruc^{n}.
2) Register " "
3) I/O " "

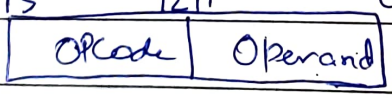
~~So based on this Instruc^{n}, the~~

Instruc^{n} Code can be of 3 types:-

1) Immediate Address
2) Direct "
3) Indirect "

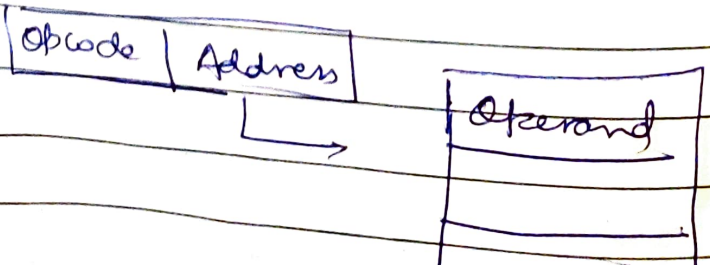
1) Immediat address → Specifies actual Operand.

Operand = Address field



2) Direct add: Address of Operand (effective address)

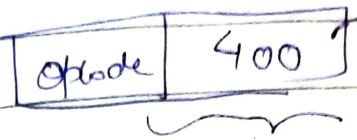
(Absolute addressing mode)

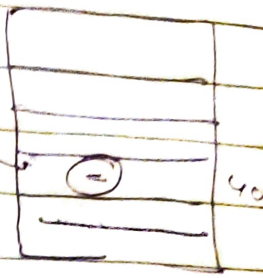


→ Actual address is given in instruction

→ use to access Variables

Ex.





जो value लिखी है वो आपका address है Operand का

Add X → Here is some value we have to load in memory

Accumulator

AC ← AC + M[X]

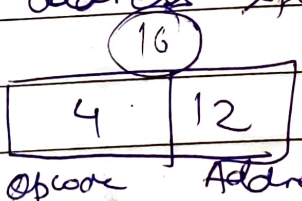
load ← LD 400

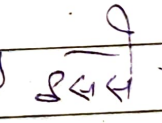
AC ← M[400]

Disadv.

- Restricted address Space

Suppose



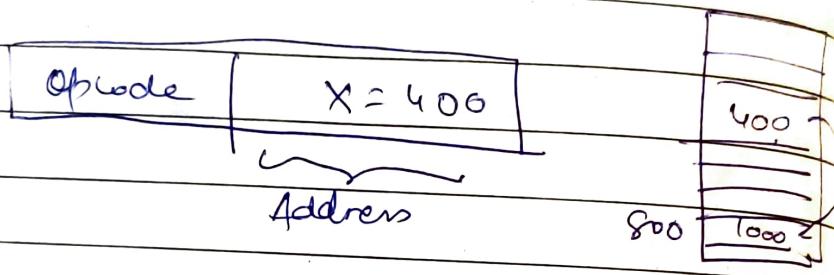


memory add करने के लिए accomodate ही नहीं पाएगा।

PAGE NO.:
DATE: / /

(3) Indirect addressing mode :-

→ The address field of the instruction specifies the address of memory locn. that contains the effective add. of the operand.



⇒ 2 memory access required.

⇒ first you ~~have~~ Check in instruc where address is written, then the no. is written in that address you visited there & from there you collected the data.

Add X

AC ← AC + M[M[X]] मतलब X की जो भी locn दी है, वहाँ पे जाओ and data वहाँ से have collected from there Again go to the memory वहाँ जाकर accumulator में data add कर दो।

PAGE NO. :
DATE : / /

Effective address = M[x] = 800

- **Processor Register**

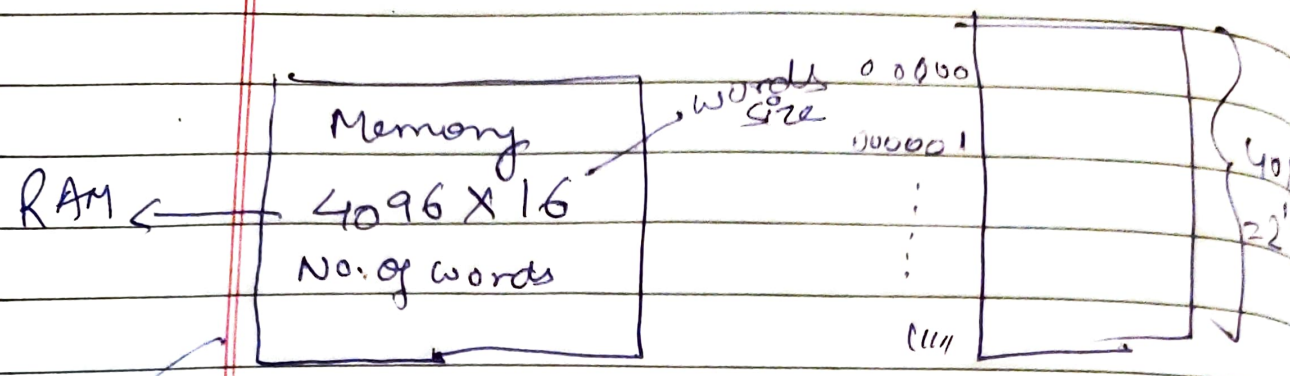
A Processor register is a small, high-speed storage loc^{n}. within the CPU that is used to store & manage data that the CPU is actively Processing. These registers are an integral part of the CPU & play a critical role in the execution of machine instruc^{n}s.

- **Effective Address :-**

The effective address refers to the address of an exact memory loc^{n}. in which an Operands value is actually present.

DATE : / /

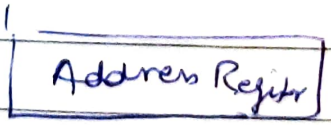
Diff. types of registers :-



⇒ Word :- It is a memory representational Unit.
1 byte = 8 bit

→ memory के अंदर हम कितने words को store कर सकते हैं, वो हैं हमारे पास 4096 i.e No. of words
यानि 2^{12} हर एक words के कितनी bits यानि सकती है। total size = 16 bits.

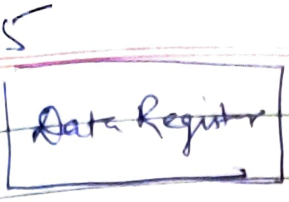
⇒ Address register :- from this we use take out data from memory.



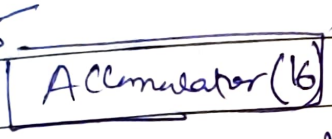
total bits = 12

⇒ Data register :- Where we use to store data.

Size of data = 16 in 1 word



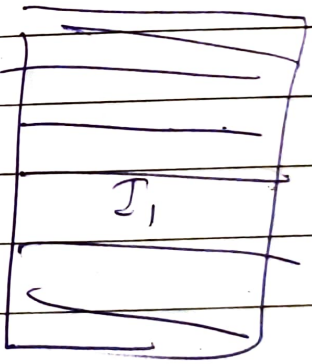
=> Accumulator :- We use To Store intermediate data.
Suppose data coming from ALU it is saved into Accumulator.



=> used for arithmetic & logical operⁿ.

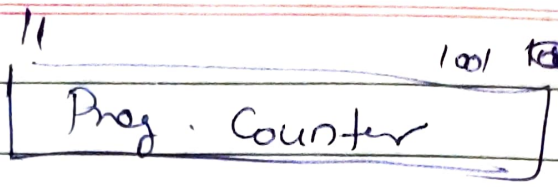
=> program Counter :- to store the address of next instruction.

Let Suppose I_1 is the instruction at '1000' address



So, in program counter, we have already fetched the I_1 instruction, So it is vacant now & next instruction is now at 1001 address I_2.

So, PC always point out those location where ~~Data~~ instruction is present, not the instruction which we have taken.

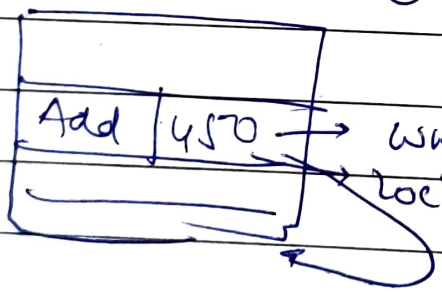


(12 bits)

=> Instruc^n. Register (IR) :-

It Contains the Instruc^n most recently fetched or executed. The fetched Instruc^n is loaded into an IR, where the OPcode & Operand Specifier is analyzed.

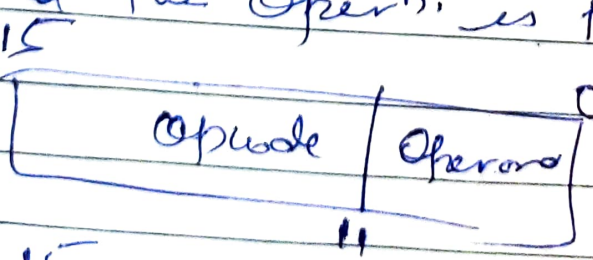
Ex:

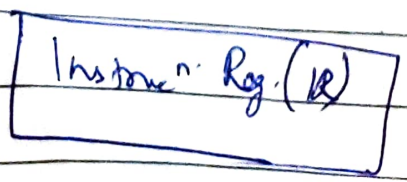


Operand : 450 is showing the data present at the 450 loc^n.

Opcode : and the Oper^n. is to be permoed

J,
edd^n, Sub^n,
mult.





PAGE NO
DATE

15 0
⇒ Temporary Reg. (TR)
(16 bits)

It is an 8-bit register which the programmer can't access at all. It is temporarily stored inside 8085 microprocessor which is 8-bit operand to the instruction set.

Adv. → Stores the information internally.
Disadv. :- users rather programmer don't have any access to it.

⇒ Input Register :- (INPR)
(8 bits)

taking i/p from devices

⇒ Output Reg. (OUTR) 8 bits

Data received

जो O/P. ALU ने दिया। और उस O/P को O/P devices को देता है, printer, monitor etc.

PAGE NO. :
DATE : / /

Common Bus Systems

A Common bus System in Computer architecture refers to a design where multiple Components within a Computer System share a single Communcn pathways, known as a bus, to exchange data & Control Signals.

This bus Serves as a Central means of Commncn, allowing diff. parts of the Computer to interact with each other.

Common bus systems are prevalent in many Computer architecture, including those found in personal Computers & embedded Systems.

⇒ Using Diagrm

There are diff. diff. register in this 16-bit Common bus System

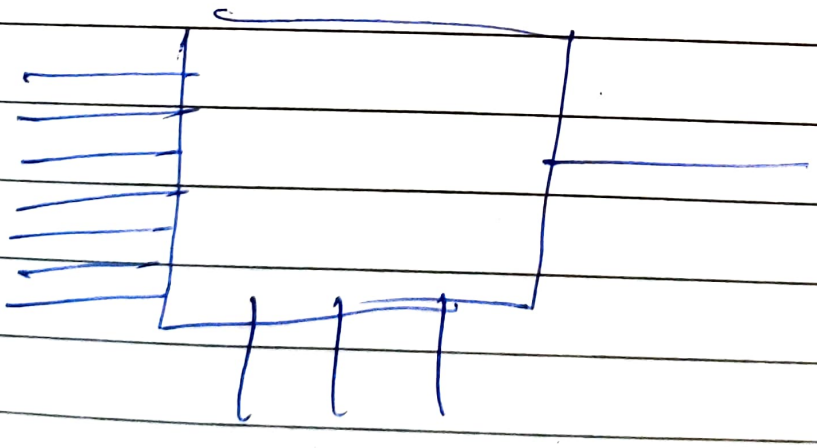
→ Address register, Prog. Counter, direct register, Accumulator, Inbut register, Instrcn register, Temporary register, Output register.

So we are connected this all register & memory unit through common bus system.

So, here we are taking 3 Select line bcz here we are connecting 7 i/p line with bus i.e, memory unit, AR, PC, DR, AC, IR, TR

So, $2^3 = 8 = 8 \times 1$

(2^n)



Note:- No of i/ps is not 8 here, only 7 is there, you can say 1 i/p is reserved. So in future if we are going to add one component then we can use there.

4096 → memory के अंदर कितने no. of slots हैं
16 → हर एक slots में कितनी bits/data को Store कर सकते हैं

PAGE NO.:
DATE: / /

→ memory Unit नहीं पर data Store किया हुआ है

Note: we have not connected input register to the bus bcz it is taking input from i/p devices like keyboard

↕ → F denotes for extra bits.

→ Here Output register is also not connected to the bus bcz, it is giving o/p to the o/p devices like monitor, printer, Scanner but it will take i/p from bus.

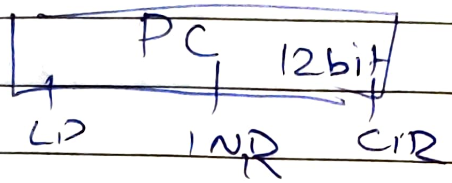
(1) memory Unit :-
Here you can see Write & Read which is a Control Value.

→ Let Suppose we will active Read here to give particular address from where to take ~~memory~~ data from memory. And after that the data is loaded onto the bus.

So, for this, i.e to load value onto bus, we will take Select

Value as 111

(2) <u>Program Counter</u> :- next instrucn to be fetched उसका address रहता पर है।



∴ we have used 12 bit address.

So, we will fetch that address & after that we have to sent that address to the address register.

Also, AR is also connected to the memory unit. So, AR will tell to the memory unit that take this address & ~~from~~ ~~data~~ data from this address to be sent to the bus. So that data can be sent to DR.

So, overall we can see from PC the operation is started

So, value '2' → 010 sent to

PAGE NO :
DATE : / /

Select line. So that the value which is present in PC will be sent to the Bus. So that the value will be available for all.

So in AR, the Control unit will get the value of load to 1.

So, the address will be sent to AR & address is of 12 bits

→ So, the same address is sent using address bus to the memory Unit.

So, in memory unit there is a decoder, which will decode the 12 bits & reach to the actual position / slot and then fetch the data present on that slot. means try to Read.

So, this data will be loaded to the bus by taking '7' → 111 as select line.

After that the data will be shared with everyone.

DATE : / /

But we are actually giving that data to DR and AC is connected to ALU (Performs some oper. like add, Subtract)

⇒ Now, that data is sent to Adder & logic and will be sent to accumulator.

⇒ from there, if accumulator wants to share that data to output register.

So, we will take value 4 → 100, so by doing this the value will be loaded to bus and after loading on bus, the same data will be available to all.

⇒ So, if we want to load that valued Output register then we will set the value of load to 1. the data will be loaded on Output.

⇒ from there that data is sent to O/P devices.

PAGE NO. :
DATE : / /

* So, you can see all load, increment & Clr is connected to CLOCK. so when clock value is activated then ~~based on~~ acc to it, it will load the data & perform clear, increment the value.

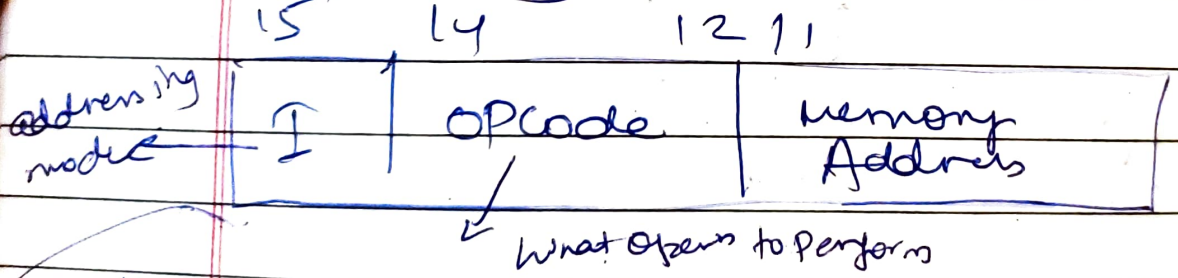
PAGE NO. :
DATE : / /

Instruction format

(1) Memory reference Instruction :

These instruction refers to memory address as an operand. The other operand is always accumulator.

It specifies 12-bit address, 3-bit Opcode & 1-bit addressing mode for direct & indirect addressing.



Ex. IR register Contains = 0001 XXXX

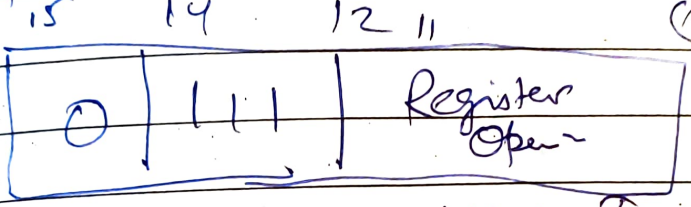
i.e ADD after fetching & decoding of instruction we find that it is a memory reference for ADD operation.

Range from 000 - 110

Opcode

(2) Register Reference :- These instructions perform Operⁿ on registers rather than memory address. The IR (14-12) is 111 (differentiates it from memory refrence) & IR(15) is 0 (differentiate it from I/P / O/P instrⁿ).

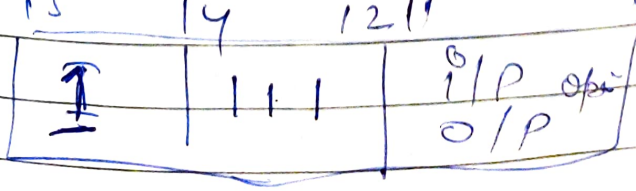
The rest 12 bits Specify register Operⁿ.



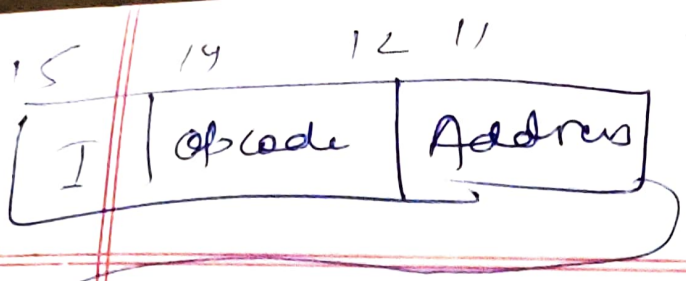
Opcode = 111, I = 0

(3) Input/Output :- These Instrucⁿ are for Communication b/w Computer & outside environment, The IR (14-12) is 111 (differentiate it from memory refrence) & IR (15) is 1 (differentiate it from register refrence Instrucⁿ).

The rest 12 bits Specifys I/O Operⁿ



Opcode = 111
I = 0



Memory reference Instructions

PAGE NO.:
DATE: / /

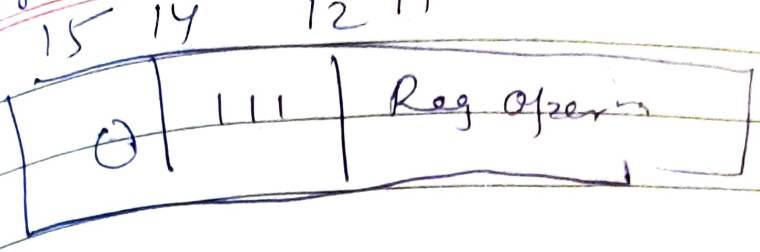
1) ~~Memory Reference Instructions~~

Symbol	Hex Code I = 0	Hex Code I = 1
AND	0xxx	8xxx
ADD	1xxx	9xxx
LDA	2xxx	Axxx
STA	3xxx	Bxxx
BUN	4xxx	Cxxx
BSA	5xxx	Dxxx
ISZ	6xxx	Exxx

⇒ I = 0, I = 1, It indicates whether it is a direct or indirect addressing. It holds the hexa code.

⇒ from 0 to 6 → direct addressing
" 8 to 15 → Indirect "

2) Register ref. Instruc



This instruction are related to the register reference instruc.

PAGE NO. :
DATE : / /

Instruction Set Completeness

A set of instruction is said to be complete if the computer includes a sufficient no. of instruction in each of the following categories:

1) Arithmetic, logical & shift instructions.
2) A set of instruction for moving information to and from memory & processor registers.

Instruction types

1) Functional Instruction

- Arithmetic, logic & shift operations.
- ADD → Add memory word to AC (Accumulator)
- CMA → Complement AC
- INC → Increment AC
- CIR → Circulate right AC & E
- CIL → Circulate left AC & E
- AND → AND

PAGE NO :
DATE : / /

CLA → Clear AC

(2) Transfer Instruct.
- LDA → Load Ac from memory
- STA → Store Content of AC into memory

(3) Control Instruct
- BUN → Branch Unconditionally
- BSA → Branch & Save return address
- ISZ → Increment & Skip if zero

(4) i/p / o/p Instruct.

- INP → Input Character to A
- Out → O/P Character from AC

Control Unit

The Control Unit is a part of the CPU. The CPU is divided into the arithmetic logic Unit & the Control Unit.

The Control Unit generates the appropriate timing & Control Signals to all the Oper^n involved with a Computer.

The flow of data b/w the processor, memory & other peripherals is controlled using the timing Signals of the Control Unit.

The main func^n of a Control Unit is to fetch the data from the main memory, determine the devices and the oper^n involved with it, & produce Control Signals to executes the oper^n.

PAGE NO.:
DATE: / /

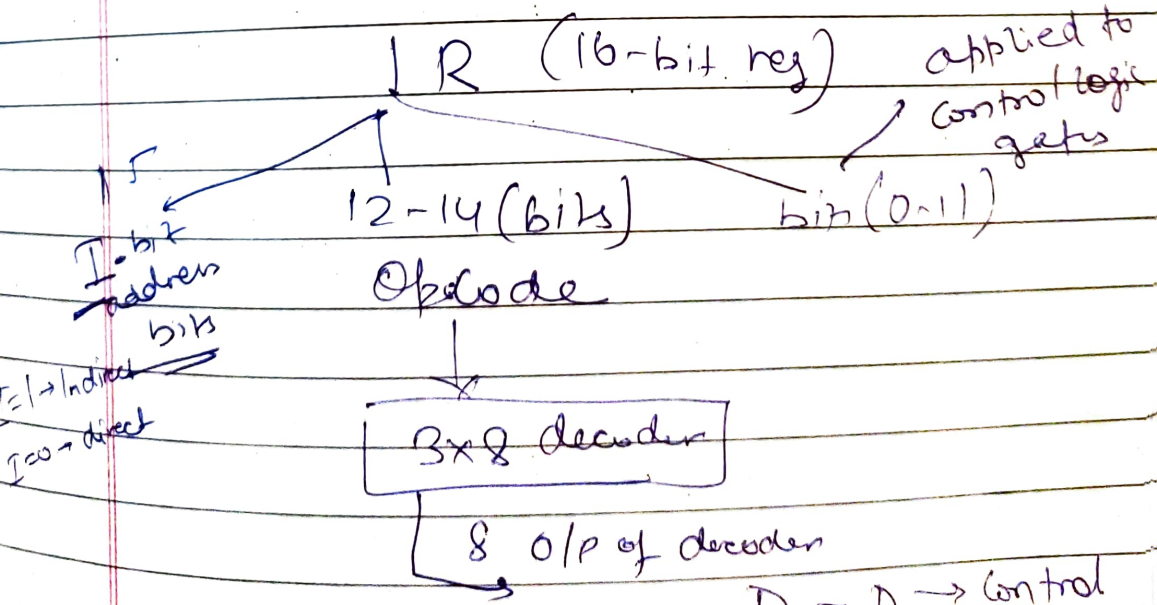
The func^{n}. of the Control Unit are :-

1) It helps the Computer System in the process of Carrying out the stored program instructions

2) It interacts with both the main memory and arithmetic logic Unit.

3) It performs arithmetic or logical oper^{n}.

Timing & Control



→ 4-bit Sequence Counter
(0-15)
↳ o/p timing signals T_0-T_{15}

⇒ 0-11 → Connected to Control logic

⇒ 15 → 9th is I (I = 0
I = 1

⇒ 12, 13, 14 → 3×8 Decoder
(0-7 o/p)

D_0 ... D_7 Connected to Control lg

⇒ 6 bit sequence counter is connected to 4×16 decoder

and o/p of decoder is 0 to 15 which is timing signals (T_0 ... T_{15}) Connected to Control logic

Note: Every timing signal will be incremented by 1.

PAGE NO. :
DATE : / /

Ex: Construct a case where sequence counter is incremented to provide timing signals.

T_0 T_1 T_2 T_3 T_4

↓

S. counter ↑

⇒ For each clock pulse the sequence counter is incrementing. Like wise, it is incrementing synchronously.

So, assume T_4 (time) Sequence counter is clear to 0 ; SC ← 0.

- If decoder o/p d_3 is active at the the time SC ← 0

So, $D_3 T_4$: SC ← 0

PAGE NO. :
DATE : / /

Instruction cycles:

An Instruction Cycle also known as a fetch-decode-execute cycle is the basic operation performed by CPU to execute an instruction.

The instruction cycle consists of several steps, each of which performs a specific function in the execution of the instruction. The major steps in the instruction cycle are:

1. Fetch :- In this, the CPU retrieves the instruction from memory. The instruction is typically stored at the address specified by PC. The PC is then incremented to point to the next instruction in memory.

2. Decode :- In this, the CPU interprets the instruction & determines what operation need to be performed.

PAGE NO. :
DATE : / /

3) Execute :- In this the CPU performs the op^n. specified by the instruc^n. This may involve reading or writing data from or to memory, performing arithmetic or logic oper^n. on data.

Fetch & Decode

Fetch:
T_0 : $AR \leftarrow PC$ (If incremented as it is going to read next Instruc^n)
T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
(Some word add. is sent to IR)

Decode:
T_2 : $D_0 \dots D_7$ Decode $IR(12-14)$
$AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$

Using diagm

Control Signal → S0, S1, S2

To achieve the follow. Connections :-

PAGE NO. :
DATE : / /

At T_0

1. Place the content of PC onto the bus by making the bus selection i/p $S_2 S_1 S_0 = 010$

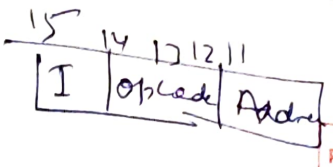
2. Transfer the content of the bus to AR by enabling the load input of AR.

At T_1

1. Enable the read i/p of memory.
2. Place the content of memory onto the bus by making $S_2 S_1 S_0 = 111$

3. Transfer the content of the bus to IR by enabling the load i/p of IR.

4. Increment PC by enabling the INR i/p of PC.



PAGE NO. :
DATE : / /

Program → Memory Unit
Sequence of inst. → IR

Step 1 Start SC ← 0
2 T_0 : AR ← PC
3 T_1 : IR ← M[AR], PC ← PC+1 (PC ↑ incremented) } fetch from memory

4 ~~T_1 : IR ←~~
T_2 :- decode opcode IR(12-14)
AR ← IR(0-11), I ← IR(15) } decode

5 T_3 :- decision
↘ Indirect
Direct

Memory reference
Register "
I/O reference

$\overline{D}_7 \rightarrow$ Decoder (0 → 7)

4 diff. modes of Instruc :-

(1) $D'_7 I T_3$: AR ← M[AR] } Indirect add. modes
D=0, I=1, T_3 Active

(2) $D'_7 I' T_3$: Nothing } Direct add.
D=0, I=0, T_3 → Active

PAGE NO. :
DATE : / /

(3) $D_7 I' T_3$: Execut a register ref. Inst.

$Q = I$, $I = 0$, $T_3 \rightarrow$ Active

(4) $D_7 I T_3$: Execute I/O instn.

$D = 1$, $I = 1$, T_3 Active

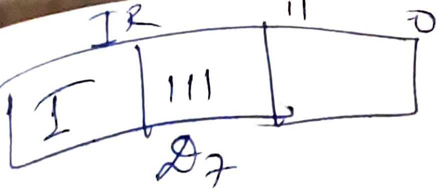
Register reference Instn:

$D_7 = 1$ and $I = 0 \rightarrow$ register ref. Inst

=> These instructions use bits 0 through 11 of the instn code to specify one of the 12 instn.

=> These 12 bits are available in IR (0-11)

=> The were also transferred to AR during time T_2.



PAGE NO. :
DATE : / /

⇒ The control funcⁿ. & microoperations for the register reference instrⁿ. are :-

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

↓ D_7 high, ↓ $I = 0$, ↘ At T_3 time

$IR(i) = B_i$ (bits in IR (0-11)) that tells the operⁿ.

↓ it will perform from 0-11

Ex $IR(1) = B_1$

⋮

$IR(11) = B_{11}$

Class
CMA → Complement AC
CME → Complement E

CIR → Circular Shift (rB_7)

↓

AC ← shr AC (right shift)

AC(15) ← E

AC 15 bit वाली है उसे E bit से fill करता है

E ← AC(0)

E की value Accumulator के 0 bit से करता है

CIL :- Circular Shift left. (rB$_5$)

AC ← shl AC

↓

Performed left shift

AC(0) ← E

Accumulator के 0 bit में Extra bit send हो जाएगी

E ← AC(15)

E में Accumulator की 15th bit आएगी

⇒ SPA : rB$_4$

if (AC(15) = 0) then PC ← PC+1

Skip if +ve

means AC(15) की value '0' है तो +ve है तो PC की value increment करती है।

⇒ SNA : rB$_3$

if (AC(15) = 1) then PC ← PC+1

Skip if -ve

means A(15) की value '1' है तो -ve है तो PC की value increment करती है।

PAGE NO. :
DATE : / /

⇒ SZA rB_2 :- If (AC=0), then

PC ← PC + 1) Skip if AC zero

⇒ SZE : rB_1 If (E=0) then

PC ← PC + 1 Skip if E is zero

⇒ HLT : rB_0 : S ← 0 (S is a Start - Stop f/f)

Halt Computer

Suppose Example

rB_{11} → means $\underbrace{I}_{r} \; \underbrace{D_7}_{B_1} \; B_{11}$, Each bit 0

0 111 1 000 0000 0000

I 7 8 0 0

Hexa Decimal Code : 7 8 0 0

Memory reference Instruction.

→ There are 7 memory-reference instructions.

→ The decoded o/p D_i for $i = 0, 1, 2, 3, 4, 5$ & 6 from the operation decoder that belongs to each instruction is included in the table.

→ The effective address of the instruction is in the address register AR & was placed there during timing signal T_2 when $I = 0$. or during timing signal T_3 when $I = 1$.

→ The execution of the memory-reference instruction starts with timing signal T_4.

→ The actual execution of the instruction in the bus system will require a sequence of microoperation. This is bcz data stored in memory cannot be processed directly.

PAGE NO. :
DATE : / /

Note: Already we have used D_7 in register reference.

So now left is $D_0 - D_6$. that's why

$q = 0, 1, 4, 5$

Total 7 Instruction. AND, ADD, LDA, STA, BUN, BSA, ISZ

(1) AND D_0 $AC \leftarrow AC \wedge M[AR]$

इन दोनों Data का AND operation करना है & result AC में store करना है।

(2) ADD D_1 $AC \leftarrow AC + M[AR]$
$E \leftarrow Cout$

Adding data of memory word in AR & AC and the Carry generated is added into E f/f.

PAGE NO. :
DATE : / /

(3) LDA D_2 $AC \leftarrow M[AR]$

loading memory word present in AR into AC.

(4) STA D_3 $M[AR] \leftarrow AC$

[Store AC]

Data present in AC is now stored in memory word in AR

Branching instruction.

(5) BUN D_4 $PC \leftarrow AR$

Data present in AR is added / transfered into PC. (Jump still)

(6) BSA D_5 $M[AR] \leftarrow PC$, $PC \leftarrow AR+1$

↓

Doing branching & returning Save address

the data in PC is Stored in a loc^n at memory word AR & incrementing the value of AR and then added into PC.

PAGE NO. :
DATE : / /

(7) ISZ D_6 $M[AR] \leftarrow M[AR]+1$

Increment & skip if zero

If $M[AR]+1=0$ then $PC \leftarrow PC+1$

~~Memory~~ add. register में जो data है उसे increment करना है।

and यदि $M[AR]+1=0$ then Prog. Counter को increment करना है skip करना है।

---

(1) AND to AC

AND logic oper. on pairs of bits in AC and the memory word specified by the effective address.

The result of the oper. is transferred to PC.

microoper.

D_0T_4 : $DR \leftarrow M[AR]$

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

This instruc. is completed at T_5 time.

PAGE NO.
DATE : / /

The Control func^n. for this Instruc^n. uses the oper^n. decoder D_0 since this O/P of the decoder is active when the Instruc^n. has an AND oper^n. whose binary code value is 000.

Two timing signals are needed to execute the instruc^n.

The clock transition associated with timing signal T_4 transfers the operand from memory into DR.

The clock transition associated with timing signal T_5 transfers the operands from AND oper^n of AC & DR to AC.

(2) ADD to AC

D_0T_4 : $DR \leftarrow M[AR]$

D_0T_5 : $AC \leftarrow AC + DR$, $E \leftarrow C_{out}$
$SC \leftarrow 0$

PAGE NO. :
DATE : / /

The same two timing signals T_4 & T_5 are used again but with Oper. decoder D_1 instead of D_0

After the instruction is fetched from memory & decoded, only one O/p of the Oper. decoder will be active & that O/p determines the sequence of microOpers that the control values during the execution of a memory reference instruction.

(3) LDA (load.) to AC

D_2T_4 : $DR \leftarrow M[AR]$

D_2T_5 : $AC \leftarrow DR$, $SC \leftarrow 0$

This instruction transfer the memory word specified by the effective address to AC.

There is no A/C to bus system there is no direct path from the bus into AC.

The adder & logic Circuit receive. inform^{n} from DR & Can be transferred into AC.

Therefore it is necessary to read the memory word into DR.

and at T_5 time SC ← 0 the data in DR is stored into AC

(4) STA ; Store AC

This instruc^{n} Stores the Content of AC into the memory word specified by the EA. Since the O/p of AC is applied to the bus & the data I/P of memory is Connected to the bus, we Can execute the instruc^{n} with one microoper^{n}.

$D_3 T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$

PAGE NO. :
DATE : / /

5) BUN :- Branch Unconditionally

This Instruc. transfer the prog. to the instruc. specified by the E.A.

Remember that PC holds the address of the instruc. to be read from memory in the next instruc. cycle.

Pc is incremented at time T_1 to prepare it from the address of the next instruc. in the prog. sequence.

This Instruc. allows the programmer to specify an instruc. o of sequence & we can say that the prog. branches unconditionally

$$D_4 T_4 : \quad PC \leftarrow AR, \; SC \leftarrow 0$$

effective add. is transfered through the common bus to PC.

The next instruc. is then fetched & executed from the memory

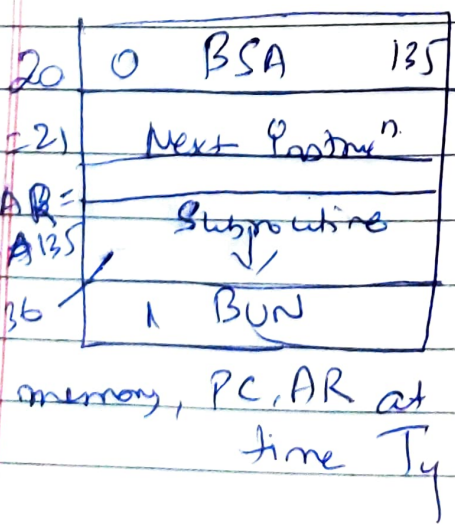
given by the new values in PC

BSA Branch & Save return address

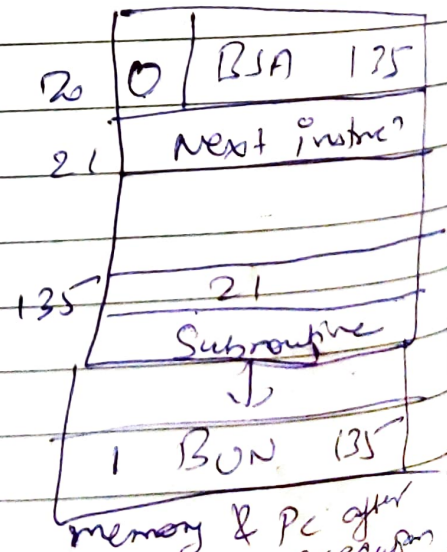
$M[AR] \leftarrow PC$, $PC \leftarrow AR + 1$

When executed, the BSA instruction stores the address of the next instruction in sequence into a memory location specified by the effective address.

The effective address plus one is then transferred to PC to serve as the address of the first instruction in the subroutine.



memory, PC, AR at time T_4



memory & PC after execution

PAGE NO. :
DATE : / /

~~first our next instruc^n is at PC = 21~~

Firstly, PC = 21 So next instruc^n will execute. So there will be some return address at PC 21

AR = 135 ~~42~~ ξ indicating i.e EA

=> In BSA, it is already we are seeing that we have to return at 135 and PC will be stored at EA.

data of

M[135] ← 21, PC ← 135 + 1
= 136

Some our next instruc^n will be continued.

---

two timing signal T_4 & T_5

decoder $D_5 T_4$: M[AR] ← PC, AR ← AR + 1

$D_5 T_5$: PC ← AR, SC ← 0

PAGE NO. :
DATE : / /

ISZ Increment & Skip if Zero

Sequence of microoperⁿ

3 timing Signals

D_6T_4 : $DR \leftarrow M[AR]$

D_6T_5 : $DR \leftarrow DR + 1$

D_6T_6 : $M[AR] \leftarrow DR$ if $(DR = 0)$
then $(PC \leftarrow PC + 1)$
$SC \leftarrow 0$