opcode → operation to perform.

OP ← reference → or which two operands to perform the operation ← variable

we want classmate

Date
Page

4/10/22.

Unit-2 Central Processing Unit.

is a intermidiate blw high level and machine

Low level lang which has some specific in pater to do programming with the hardware.

Assembly Language –

assembler converts to machine level lang.

machine instruction are represented by the patterns of 0's and 1's such patters are foud to be difficult to d while writing programs. therefore symbolic names are used to represent the patterns.

When writing programs for a specific computer architechure, some ~~memonics~~ mnenocics such as ~~add~~ ADD, MOV, MUL are used

A complete set of such symbolic names and rules for their use constitue a programming languate reffered to as ~~an~~ Assembly language

Programs written in an assembly language can be translated into a sequence of machine instructions by a program called an assembler. The assembler program is a part of system software

Elements of Machine Instructions

Each instructions must contain the information by the processor for execution-

R- register →
↳ smallest unit of storage in the processor

The elements are as follows -

- Operation Code -
  The operation is specified by a binary code, known as operation code or opcode.

  Specifies the operation to be performed (eg - ADD, I/O).

- ~~Result~~ Source Operand Reference -
  The ~~operation~~ on which the operation will perform.

- Result Operand Reference -
  The operation may produce a result that will store in result

- Next instruction Reference -
  This tells the processor to fetch the next instruction.

Instruction Representation -

Whithin the computer, each instruction is represented by a sequence of bits. The instruction is divided into fields, corrosponding to the constituent elements of the instruction.

Operation Code	Operand Reference	Operand Reference

Common Examples-

ADD --	Add
SUB --	Subtract
MUL--	Multiply
DIV --	Divide
LOAD--	Load data from memory.
STOR --	Store data to memory.

for Eg - Consider the instruction in high level
~~ADD R,Y~~ X = X+Y

The assembly lang code for above instions can be writt·en

~~AXX~~ ADD R,X
X=6 ,Y=9·
x = x+y·

LOAD Y,R
LOAD X,R1
ADD R,R1
STOR R1,X

OR

LOAD Y,R
ADD,R,X

Smallest unit of the Processor

7/10/22

→ Processor clocks. (electronic component)

Processor circuits are controlled by a timing signal called clocks

The clock defines regular time intervals called clock cycle let P be the length of one clock cycle clock rate $R = 1/P$

clock cycle ⇒ Time interval

To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps such that each step can be completed in one cycle

$T = (N * S)/R$

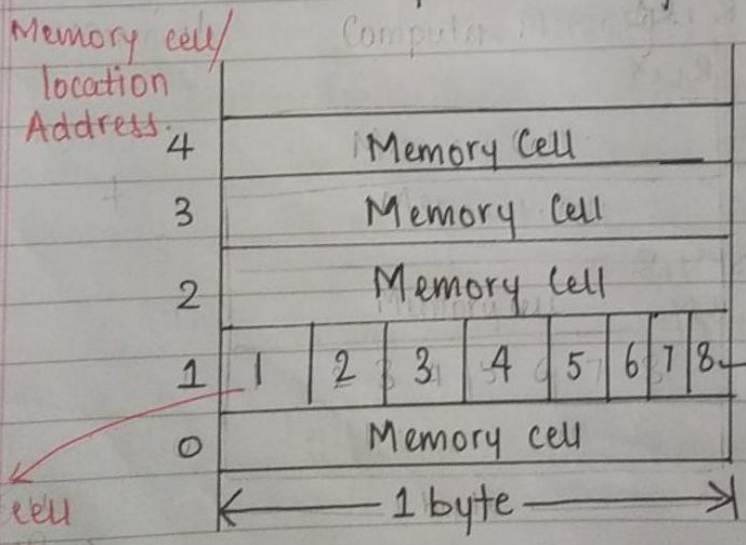
computing time / processing time

N → no. of instructions to be executed.

S → no. of avg steps.

H_1 →

Computer Memory.



→ Byte Address Ability.

The three basic information quantities are bit, byte and word.

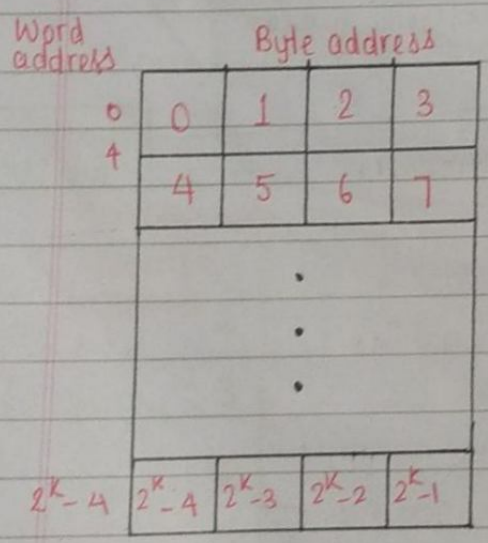
Byte is always 8 bits, but the word length varies from architecture to architecture. Therefore it is impracticle to assign distinct address to individual bit location in the memory

In most model computers, the assignment of a location of memory is done with help of byte and hence addressing the memory location through byte is called as byte address ability

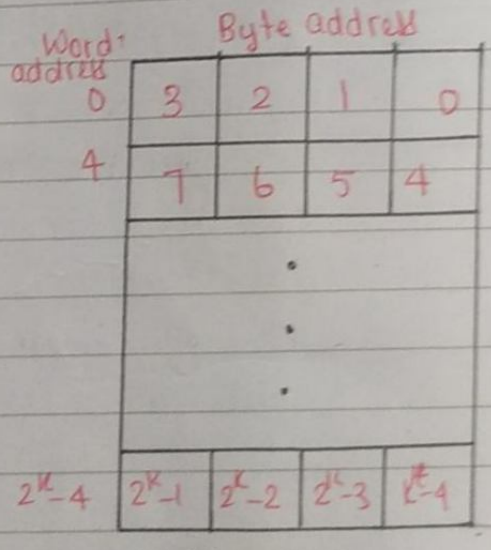
There are two ways that byte address can be assigned that are -

a) big endian.

d) little endian.



a) Big endian assignment



b) little endian

Addressing Modes.

1) The different ways in which the location of an operand is specified in an instruction are reffered to as addressing Modes

2) There are addressing modes depending upon the use of variables, constants, data structure etc.

3) The various addressing modes are immediate addressing modes, register, absolute (direct), indirect, index, auto increment and auto decrement etc.

10/10/22

Implementation of variables and constants.

Variables and constants are the simplest data that are found in almost every computer program

In assembly language a variable is represented by allocating a registor or memory location to hold its value

For the implementation this following addressing modes are used

1) Immidiate Addressing Moole
2) Register Addressing Mode
3) Absolute (Direct) Addressing Mode.

1) Immidiate Addressing Mode
In this addressing mode the operand is given explicitly in the instruction.
For eg - MOV #200 , R0
In above eg the value 200 directly assigned to register R0

A common convention is to use # before the value to indicate that this value is to be used as an immidiate operand.

2) Register Addressing Mode
In this addressing mode, the oprand is the contents of processor register , the name of a register is given in the instruction. ~~for eg~~
For eg - MOV R , LOC

3) Absolute Addressing Mode (Direct Mode)
The operand is in memory location and the address this location is given explicitly in the instruction Such mode is called as absolute Mode (Direct Mode)
For eg ~~LOC~~ MOV . LOC , R

Implementation of Pointers
For implementation of Pointers the indirect addressing mode is used

4) Indirect addressing mode
Indirect addressing Mode ⟹ The effective address of the operand is the contents of a register or memory location whose address appears in the instruction

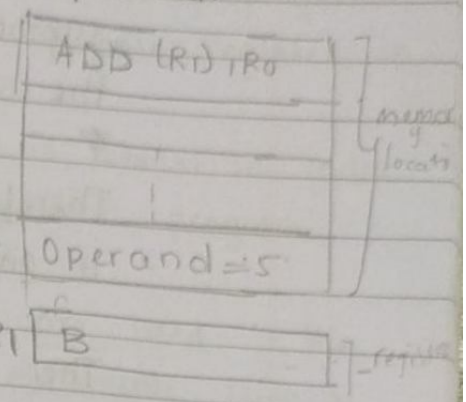
other than that it always represents the memory location.

as processor register

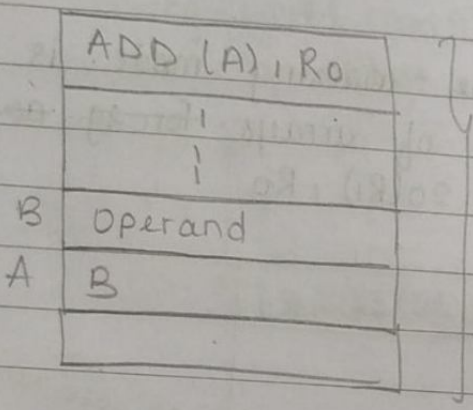
In other words, the address of memory location or the register is given in the instruction, in which the address of operand is specified.

For eg ADD (R1), R0.

act as pointer indirectly holds the address pointing to the of the variable value



Eg ADD (A), R0



→ Implementation of Arrays.

→ Index Addressing Mode.

The addressing mode in which the address of the operand is obtained by adding the content of the register to it's constant value.

This address is also called as effective address of the operand.

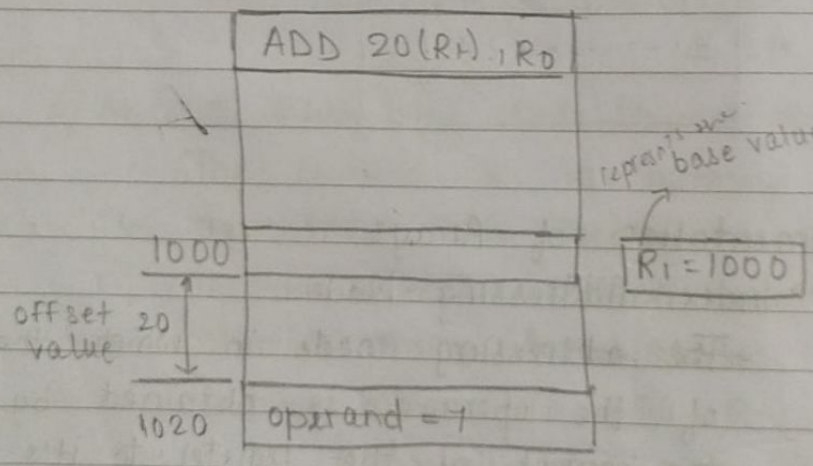
The register used may be either a special register provided for this purpose or it may be

any one of a set of a general purpose register in the processor and it is reffered to as an index register. Therefore index mode symbolically represented as X(Ri), where X denotes the constant values and Ri is the register.

Therefore, the effective address of the operand is given as EA = X+Ri

The contents of index register are not changed in the process of generating the effective address

Generally, the Indexing mode is used in the implementation of arrays ~~for eg~~ ~~ADD (20~~
For eg ADD 20(R1), R0



14/10/22

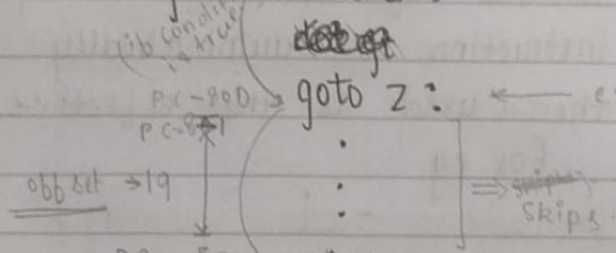
Relative Addressing Mode

The addressing Mode in which the address of the operand is determined by using program counter in place of general purpose register is called relative addressing mode.

This addressing mode is similar to index mode but the difference is that program counter is used.

Such addressing modes are used in the implementation of jumping statements such as go to statement

Eg- if (condn) ← execute



In above eg, the content of Program counter Eg- will be incremented as per the jumping statements present in the program it means that only the contents of Pc is related with itself and Hence it is relative addressing Mode

→ Auto ~~Intt~~ Increment & Auto Decrement Addressing Mode

The effective address of the operand is the contents of the register specified in the ~~regist~~ instruction, after accessing the operand the contents of this register is automatically incremented

for eg Ri.+

i++

count++

~~Simil~~ Ly, in auto decrement mode, the contents of the register specified in the instruction are automatically decremented and then used as the effective address

For eg

--i

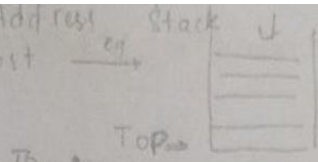
-÷ count.

→ Instruction Types

Types of Instruction.

Types
- Data Processing: Arithemetic and logic instruction
- Data Storage: Movement of data into or out of register and or memory locations.
- Data movement: I/O instructions.
- Control: Test and branch instruction.

⇒ Zero Address Instruction



dissmate
Date
Page

PUSH → insert element
POP → delete the

Instruction Format

→ Three Address Instruction

Pipeline → implement in the execution of instruction

Pipeline is the process of working with different operation unit at same time.

Pipline is a process of arrangement of hardware elements of the CPU such that it overall performance is increased.

Simultaneous, execution of more than one instruction takes place in a piplined processor

Without pipelining = 9/3 hrs = 3hrs.

EBP ||||||| = 3hrs

||| EBP ||| = 3hrs

||| ||| EBP = 3hrs

↳ Total = 9 hrs.

with pipeline → 5 1/2 hrs = 1.67 hrs

EBPII =
IEBPI =
IIEBP =

→ Total → 5 hrs

17/10/22

→ Instruction Pipeline.

Instruction Pipeline process is processor which executes different instructions belonging to a process in it's different stages.

4 stage Pipeline
- Fetch → instruction get Fetch from memory
- Decode → decode the instruction and fetch the operand/ it input instruction is the output logical Arithmetic.
- Execute → inst to get executed
- Write Back → resultant operant stored

Buffer - Temperory storage

→ 4 Stage Pipeline

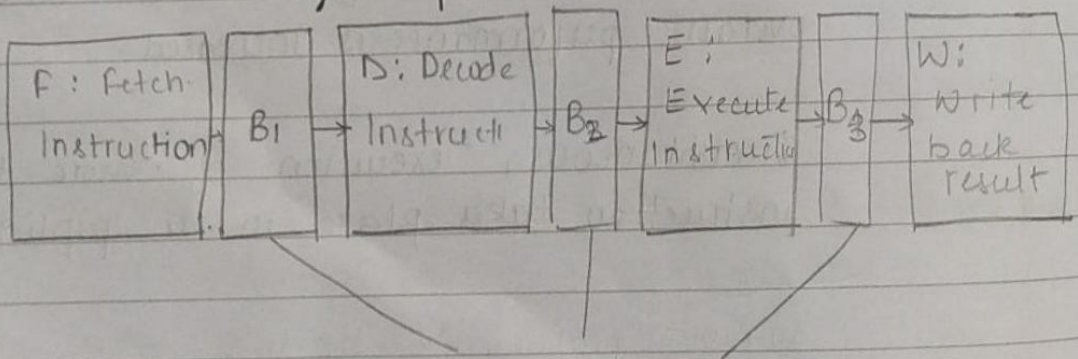


Fig- ~~hor~~ Hardware ~~H/W~~ orginsation of 4 stage pipeline

whichever stage take the highest time execute then th it will be concided as clock cycle.

Case 1 – Without using pipelining.

Instructions \ clock cycle →	1	2	3	4	5	6	7	8	9	10	11	12
I_1	F_1	D_1	E_1	W_1								
I_2					F_2	D_2	E_2	W_2				
I_3									F_3	D_3	E_3	W_2

⇒ 12 clock cycle.

Case 2 – With using Pipeline.

Instruction \ clock cycle	1	2	3	4	5	6	7	8	9	10
I_1	f_1	D_1	E_1	W_1						
I_2		f_2	D_2	E_2	W_2					
I_3			F_3	D_3	E_3	W_3				

⇒ 6 cycles.

→ Stalling.

I_1	F_1	D_1	E_1	W_1						
I_2		F_2	D_2			E_2	W_2			
			F_3	D_3				E_3	W_3	

Fig Explaination of fig a ~~Interstage buffer~~ Hardwar organisation.

The above four stages pipeline has four units discussed as follows -

1) Fetch
   This unit reads the instruction from the memory

2) Decode:
   This unit is responsible for decoding the instruction and fetching the source ~~code~~ operar

3) Execution Unit'
   This unit execute the instruction

4) Result Unit.
   This unit is responsible to write back result to register or to a memory.

All these units are connected to each other by buffer and these buffers are used to hold the information for sometime till the unit is free to perform the next task.

Explaination of Pipeline C

Each stage is in pipeline is expect to complete its operation in one clock cycle. Hence the clock ~~provided~~ period should be subbiciently long to complete the the task to be completed the task being performed in any stage.

In different units require different amounts of time the clock period must allow the longest task to be completed.

A unit that complete it's task early are ideal for remainder of clock. Hence, this pipelining is most effective in improving the performance. If the task being performed in different stages at same time.

In above fig, there instructions require 12 clock cycle to complete its execution whereas in fig by implementating instruction pipeline, there instruction require 6 clock clycle to complete it's ~~exec~~ execution.

→ Pipeline Performance.

It is assumed that the pipelined processor completes the processing of one instruction in each clock cycle which means that the rate of instruction processing is 4 times that of sequential operation.

But for a number reasons it may happen that one of pipeline stages may not be able to complete it's processing task for a given instruction in the alloted time.

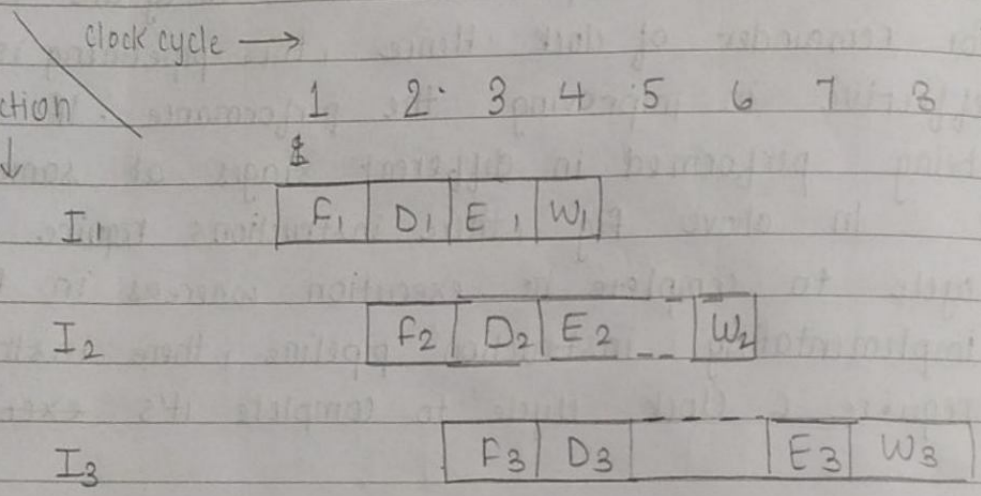
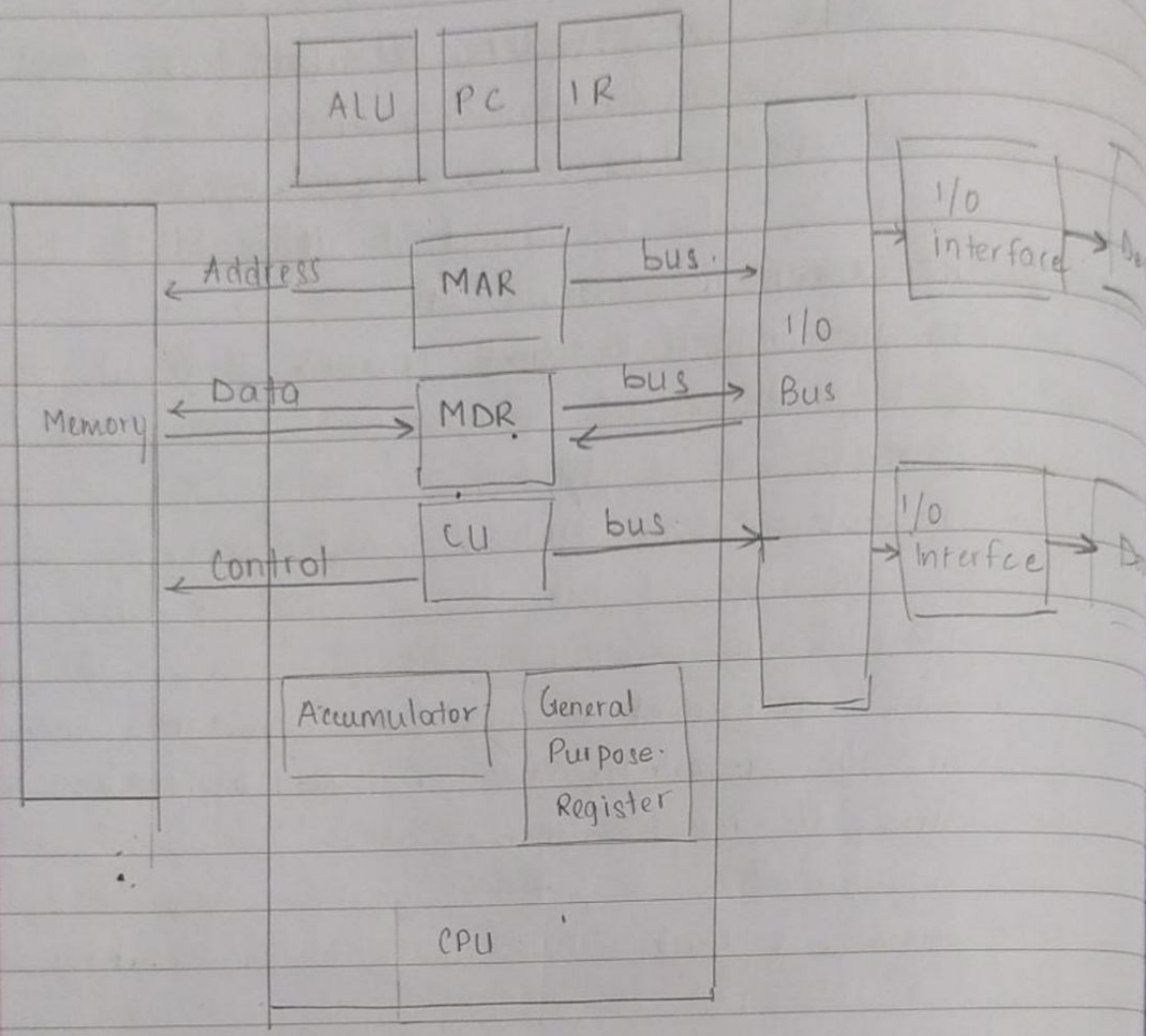


Fig Effect of an operation taking more than 1 cycle.

In the above fig, instruction I_2 require 3 clock cycle for it's execution from clock cycle 4 to 6 and therefore the execution of instruction I_3 extended by 2 clock cycle and hence these 3 instruction require total 8 clock cycle for it's instruction.

→ Basics of Processing Instructions.



The above fig shows basic processing unit (CPU) the fundamental component present in the CPU such as

ALU Arithematic
CU Control Unit

Along with this two unit there are special and general purpose register such as MABR, MDR, PC, IR, are special registers. Whereas some general purpose registers are also required All this component of CPU are

connected with each other by means of Bus.

The Bus are of 2 types
1) Single bus organistion
2) Multiple bus organisation.

1) To execute a program, the processor fetch one instruction at a time performs the operations specified

2) Instructions are fetched from sucessive memory location until a branch or jump instruction encounter

3) The processor keep track of the address of memory location containing the next instruction to be fetche using Program Counter (PC).

4) After fetching an instruction the contents of PC are updated to point to the next instruction in the sequence.

5) At the same time the register MAR sends a memory location to the ~~mo~~ memory and the register MDR brings that instruction or data from the memory location specified by the MAR.

6) Once the instruction has been fetched completely it transfers to a special register called as instruction register (IR).

4) The register IR with the help of control unit decode that instructiond and accordingly the generated the control signals.

Single Bus Structure / Organisation

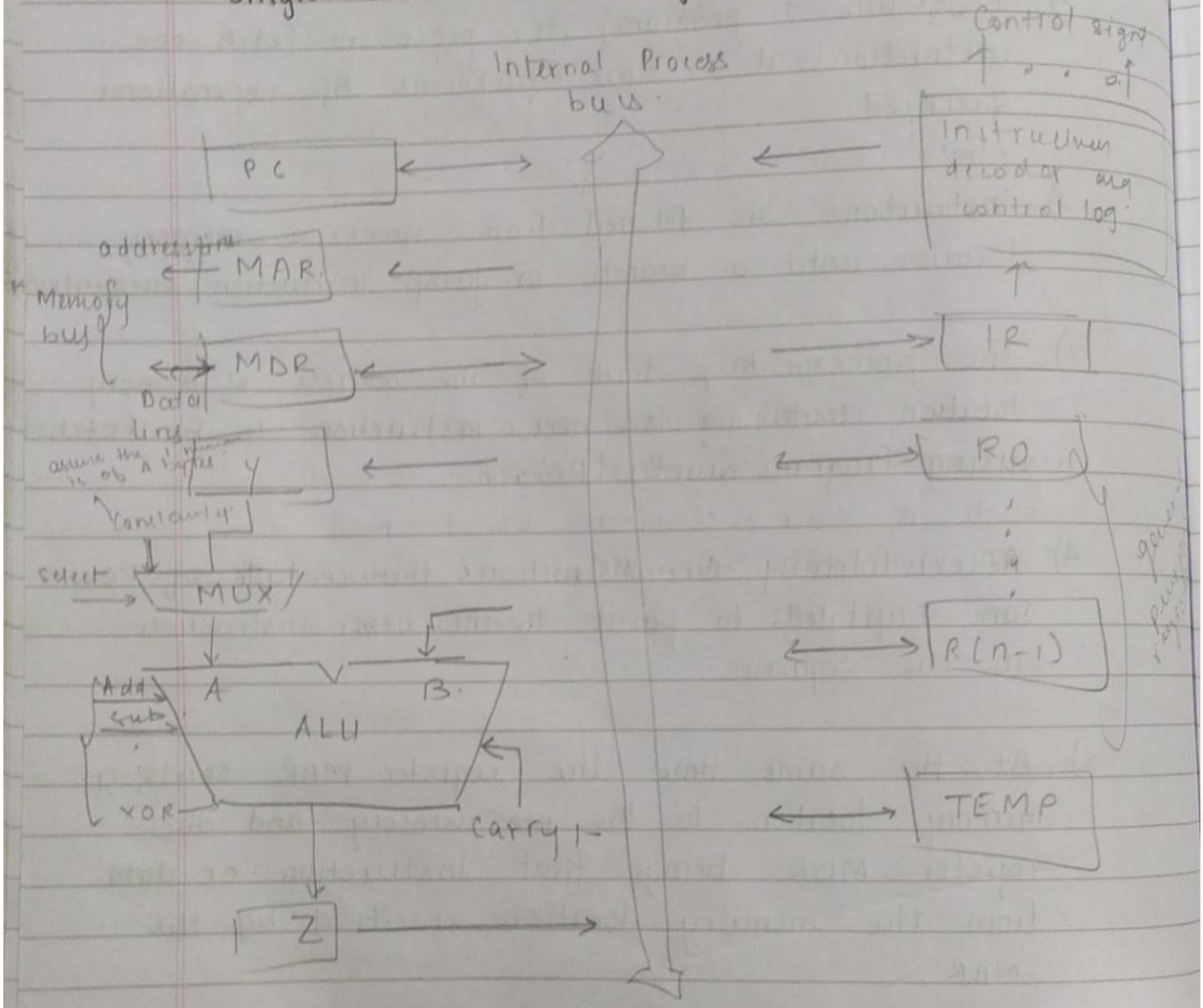


Fig - 1 - Single

1) The above figure shows a single common bus through which all the componentes of the processor such as various registors CU & ALU are connected This is the internal bus of the processor and it is diff. from external bus

2) The PC points to the ~~firs~~ current inst. to be executed, and pass on that memory location through the bus
The MAR then carries this memory location of the inst. whereas MDR is used to carry the data from respective memory location and write back the result computaon into the memory and Hence there is a 2-way arrow to MDR

→ ~~Multiple Bus Organisation~~

3) After passing the inst. to IR, the contents of PC get incremented as PC ← PC + 4 because one memory word is assume to be of size 4 bytes

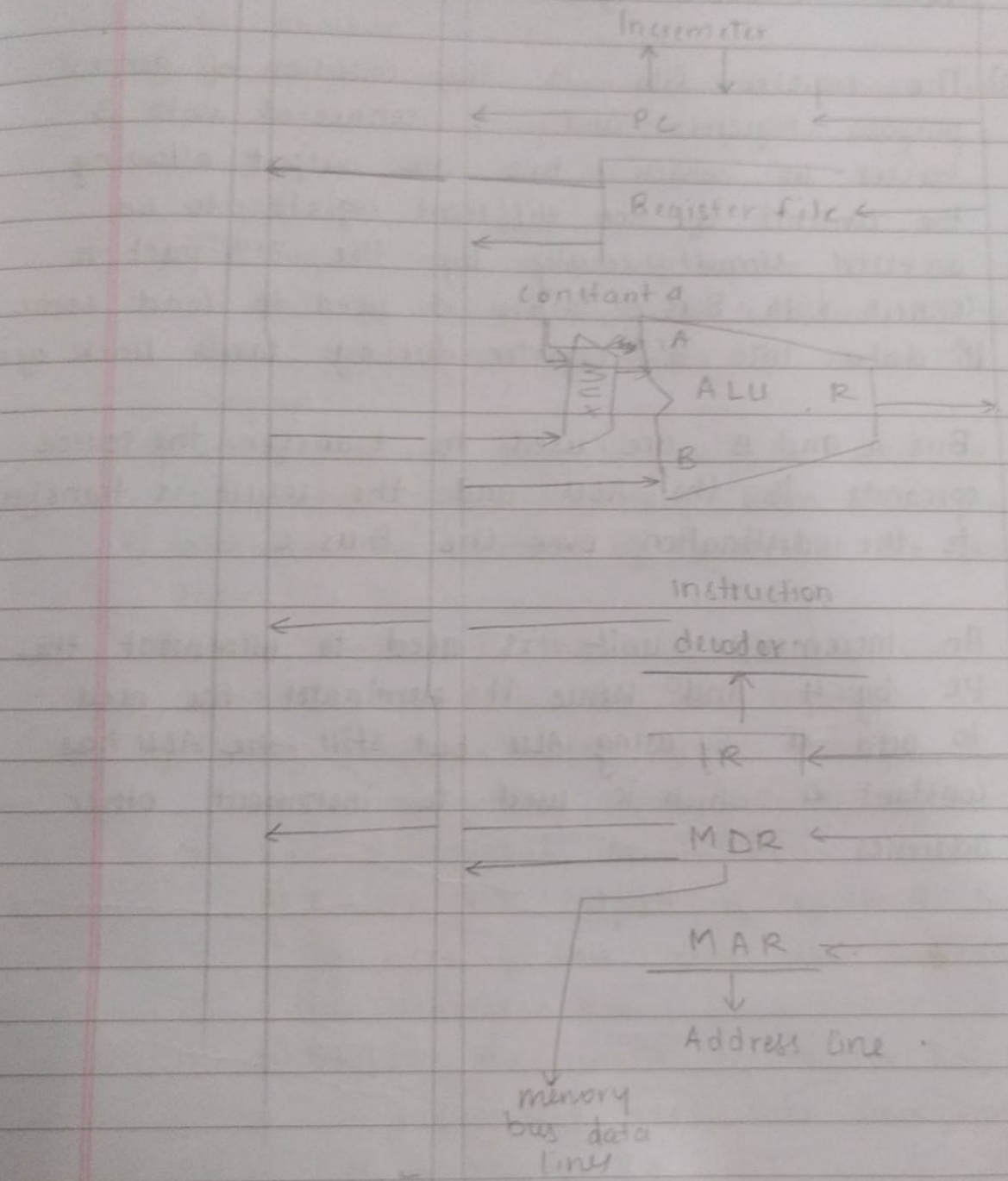
4) R0 to --- R(n-1) are general perpose register used for execution in ALU

5) Simillarly Y, Z, Temp are temperory storage register used during the execution of sub instructio

6) When instruction executed complelely or execution progresses, data & are transfered from one register to another often passing through the ALU to perform various arithematic or logic operations

7) Finally the result gets store back to the memo with the help of ~~a~~ register MDR.

→ Multiple Bus Organisation.



1) A three bus structure is used to connect the different components inside processor / CPU as shown in above fig.

2) The register file is the collection of general purpose registers and it connected with 3 busses in which two are output allowing the contents of two diffferent register to be accessed simultaneously. By the 3rd part is connect with Bus C whic is used to load some data into 3rd register during same clock cy

3) Bus A and B are used to transfer the source operands to the ALU and the result is transfer to the destination over the Bus C.

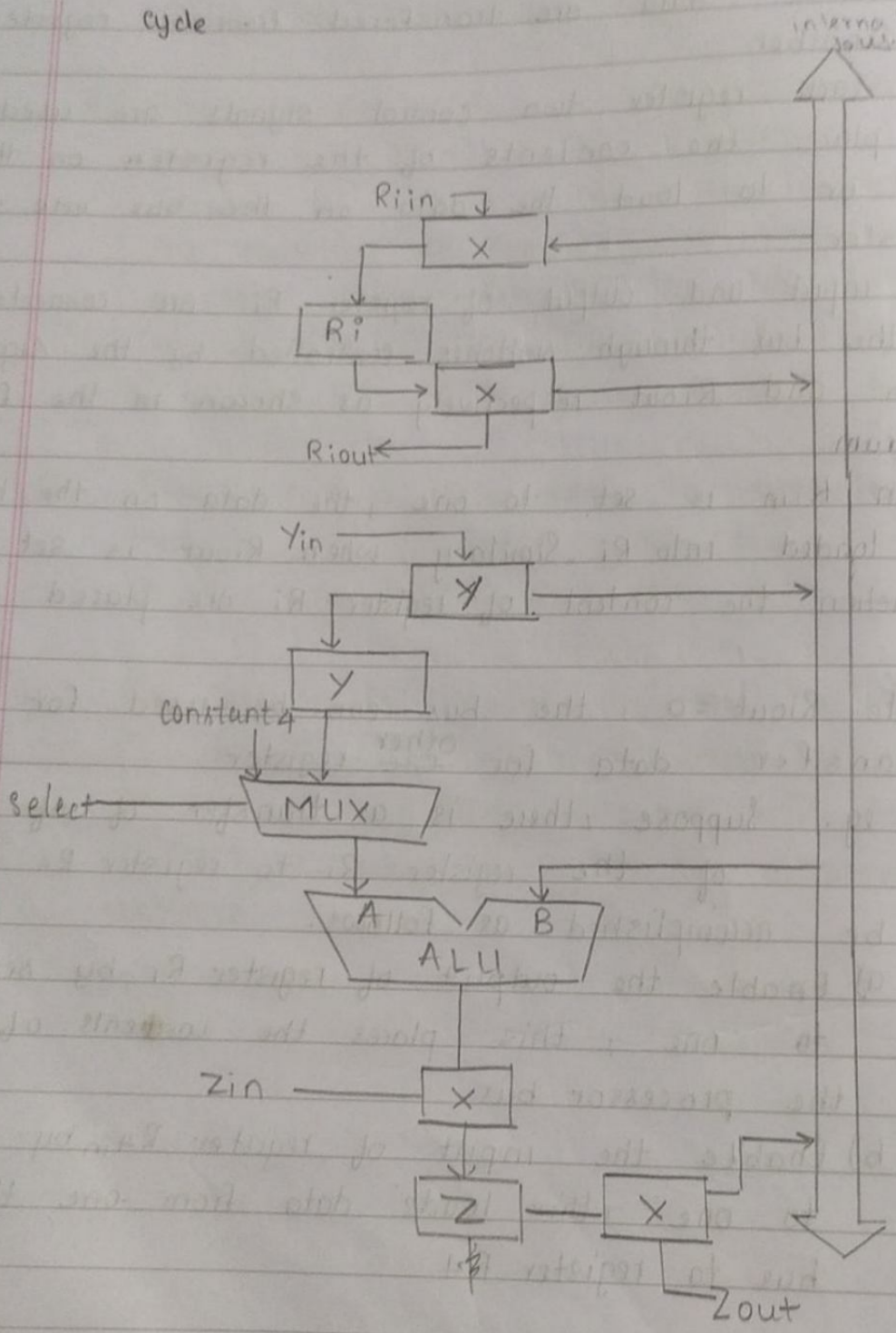
4) An Incrementer unit is used to increment the PC by 4 and Hence it eleminates the need to add 4 by using ALU but still the ALU has constant 4 which is used to increment other addresses.

Register Transfer.

1) Instruction executation a sequence of steps in which data are transfered from one register to another
2) For each register two control signals are used to place the contents of the register on the bus or to load the data on the bus into the register
3) The input and output of register Ri are connected to the bus through switches controlled by the signal Riin and Riout respectively as shown in the following diagram.
4) When Riin is set to one, the data on the bus are loaded into Ri. Similary, when Riout is set to one
5) ~~Instruction~~ the content of register Ri are placed on the bus.
5) While Riout = 0, the bus can be used for ~~transp~~ transfer data for ~~one~~ other register.
6) For eg. Suppose, there is a transfer of ~~of~~ the of the register R_1 to register R_4 this can be accomplished as follows.
   a) Enable the output of register R_1 by setting R_{1out} to one, this places the contents of R_1 on the processor bus.
   b) Enable the input of register R_{4in} by setting to one, this loads data from ~~one~~ the processor bus to register R4

All the operation and data transfers within the processor take place within time period defined by processor clock.

The control signal that handle a particular are transfer are generated at the start of the clock cycle



Size of on memory cell => 4 bytes.

18/10/22.

1). Fetching a word From Memory.

Q) How a word get fetch from the memory?

→ 1) To fetch a word for information from memory, the processor has to specify the address of the memory location where these information is stored and request a read operation.

2) This applies whether the information to be fetched represents instructions in a program or an operand specified by an instruction. ~~in~~

3) ~~a program~~ The processor transfers the requirment address to the MAR, those output is connected to the ~~same time~~ address lines of the memory bus, at the same time the processor uses the control line of the memory bus to indicate that a read operation is needed.

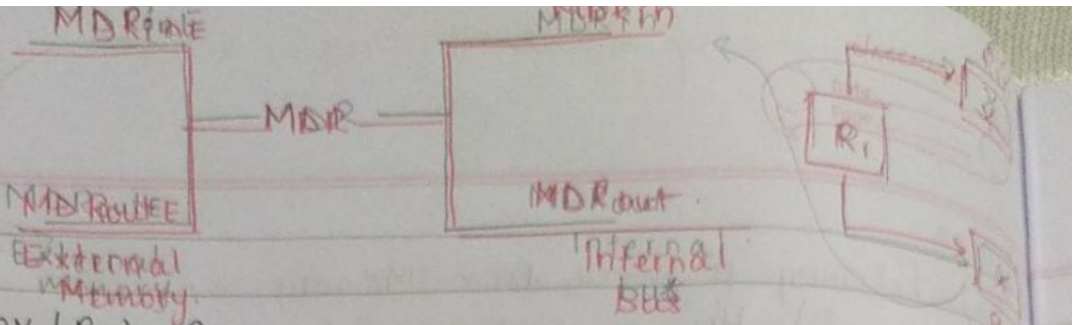
4) When the requested data are received from the memory they are stored in register MDR, from where data can be transfered to ~~an~~other register in the processor.

5) During memory read and write operations, the timings of the internal processor operations must be coordinated with response of address device on the memory bus.

6) The processor complete one internal data transfer in one clock cycle.

7) The processor ~~wit~~ wait untill it receives and indication that the requested read operation has been completed.

For such indication a control signal called Memory Function Completed (MFC) is use for the purpose.



Eg
↳ MOV (R1), R2

The action needed to execute this instruction is described as follows -

a) 1) MAR ← [R1]
2) Read operation on Memory bus.
3) Wait for MFC response from memory.
4) Load data to MDR to memory bus.
5) R2 ← [MDR]

The above three steps of memory read operation can be described by signals being activated are as follows.

Control sequences

a) 1. R1out, MARin, Read.
2. MDRinE, MFC → Memory function completed
3. MDRout, R2in.

2) Storing a word in Memory.

1) Writing a word into a memory location follows the same procedure like read operation

2) The desired address loaded into ~~mar~~ MAR, then the data written are loaded into MDR. at the same time a write operation get started.

3) After writing a comple word into a memory, MFC control signal get generated.

For eg- MOV R_2, (R_1).

The steps or action needed are given as follows-

1) MAR $\longleftarrow$ $[R_1]$
2) MDR $\longleftarrow$ R_2
3) Start write operation.
4) Wait for MFC operation.
5) ~~[MAR] $\longleftarrow$ MDR.~~

Control Sequences.

1) Riout, MARin
2) Rout, MDRin
3) MDRoutE, MFC.

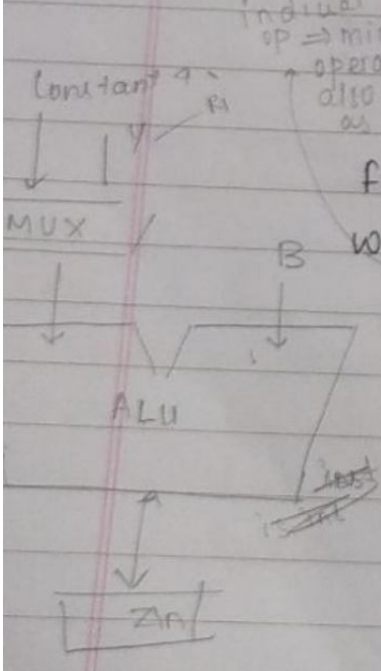
Execution of a complete Instruction

The execution of any instruction requires some sequence of operation like fetching a word from memory. Perform the arithematic operation and store back the result

Consider the inst. ADD (R_3), R_1 which adds the contents of memory location pointed to by R_3 with the contents of register R_1 ~~write~~ & stores back the result in R_1

Executing this inst. requires the following steps:

a 1) Fetch the instruction.
b 2) Fetch the ~~first~~ operands.
c 3) Perform the addition
d 4) load the result into R_1



for above steps the control sequences are written as follows.

a
1) PCout ; MARin , Read, Select4, Add 4,
2) Zout , PCin , Yin , MFC
3) MDRout , IRin
b 4) R3out , MARin , Read.
c 5) R1out , Yin , MFC
c 6) MDRout , Select Y , Add , Zin.
d 7) Zout , R1in , End.