COD

Computer Organisation and Design (Lovely Professional University)

UNIT-02

Computer Organization:

PAGE
DATE 01 02 23

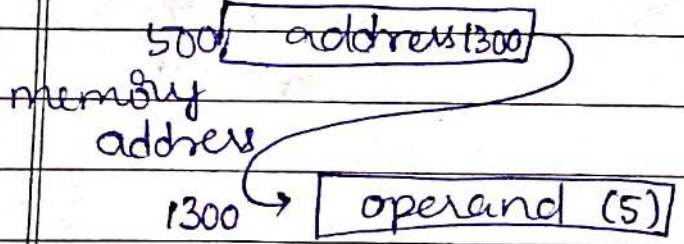
→ Organization of Computers is defined by
* Internal Registers * Timing & Control Structure
* Set of instructions that it use.

→ Modern Processor is a very Complex device.
* Many registers * Multiple arithemetic units
* Ability to Pipeline many consecutive operations.

→ A Basic Computer Contains two components
* Memory and * Processor.
→ Memory has 4096 words in it (RAM). $2^{12} = 4096$
→ Each word is 16 bits long * 12 bits are required

Program – A Set of Machine instructions (micro-operation)

→ INCrement (no address) – immediate instruction.
→ 0 - Direct addressing – Instruction with address along with data.
→ 1 - Indirect addressing – Second part of instruction designates an address of a memory in which the address of the operand is found.



Effective address.

PAGE
DATE 02 02 23

→ Accumulator - General Purpose Register.
→ Effective address - Address of the operand.

→ There are 8 types of Registers:
* Program Counter (PC) : Contains address of the next instruction to be executed
→ There is no extra bits → It is of 12 bits.

* Address Register (AR): Track of what locations in memory it is addressing. → 12 bits.

* Data Register (DR): operand is found, using either direct or indirect addressing it is stored in DR.
→ 16 bits. (operand is stored in DR)

* Accumulator: Data + Processor register. → 16 bits.
* Temporary Register (TR): To store the temporary Storage register / Temporary data
→ 16 bits.

* Input Register (INPR): Input device → 8 bits
* Output Register (OUTR): Send to output device → 8 bits.

PAGE
DATE

Common Bus System:

only for a basic computer

→ Basic Computer: 8 registers, a memory unit & a control unit.

→ The registers in the Basic Computer are connected using a BUS.

→ Output of 7 register and memory connected to input of bus.

→ 3 Select lines S_2, S_1, S_0 will be there.

→ LD(Load) Input is enabled, recieves the data from the bus (Next Clock Pulse).

	S_2	S_1	S_0	Register
0	0	0	0	X
1	0	0	1	AR
2	0	1	0	PC
3	0	1	1	DR
4	1	0	0	AC
5	1	0	1	IR
6	1	1	0	TR
7	1	1	1	Memory

Data transfer from
Bus → Reg ⇒ Load
Reg → Bus ⇒ select lines

ALU - Arithmetic logic unit

* AC ← DR ; DR ← AC is done using transfer through ALU. and Load of AC is high.

PAGE
DATE 03 02 23

Q. The output of four registers (R_0, R_1, R_2 and R_3) are connected through 4 to 1 line multiplexers to the inputs of the fifth register.

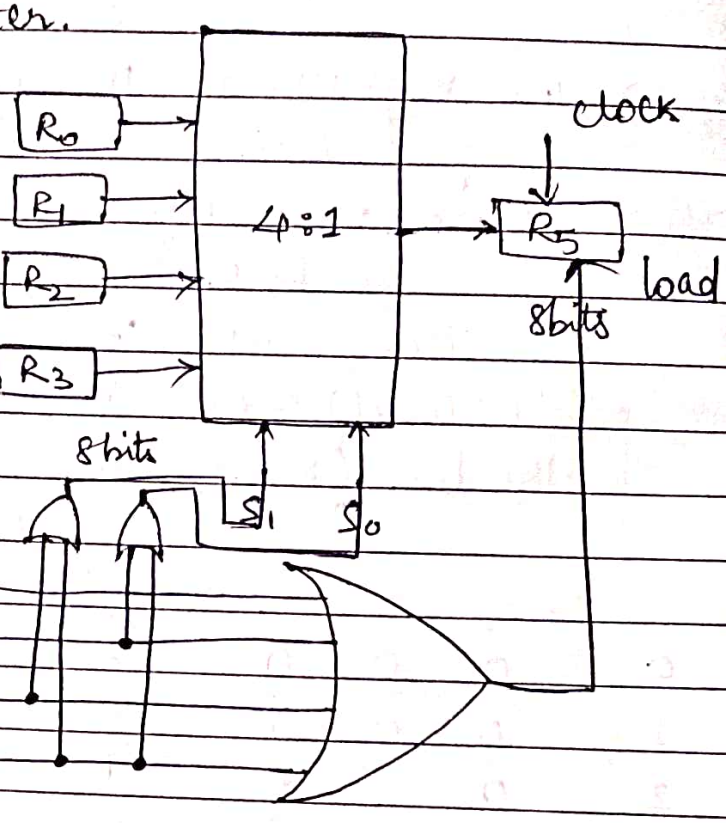
Sol:

$T_0 : R_5 \leftarrow R_0$

$T_1 : R_5 \leftarrow R_1$

$T_2 : R_5 \leftarrow R_2$

$T_3 : R_5 \leftarrow R_3$



$T_0T_1T_2T_3$	S_1	S_0	R_0	load
0000	X	X		0
1000	0	0	R_0	1
0100	0	1	R_1	1
0010	1	0	R_2	1
0001	1	1	R_3	1

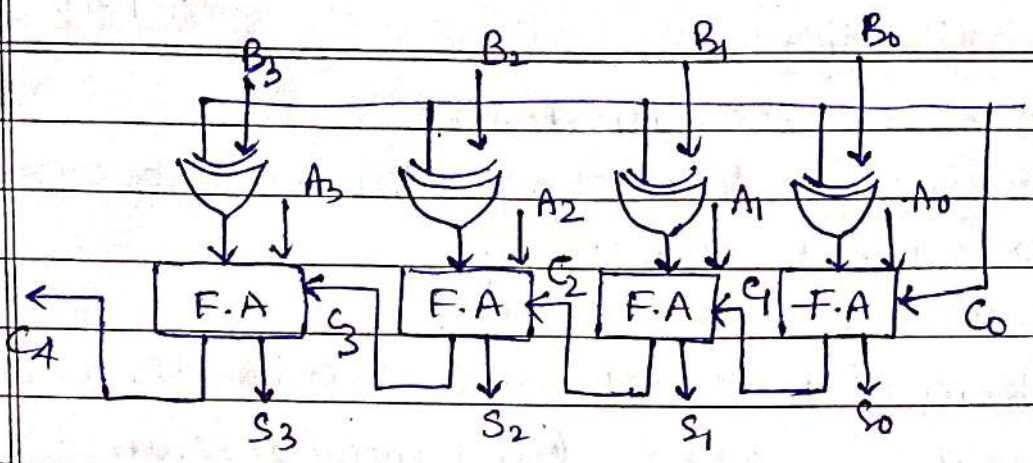
$S_1 = T_2 + T_3$

$S_0 = T_1 + T_3$; $load = T_0 + T_1 + T_2 + T_3$

Q. The adder-Substracter circuit has the following values for the input mode 'M' and data inputs 'A' and 'B'. In each case, determine the outputs: S_3, S_2, S_1, S_0 and C_4.

M	A	B
0	0111	0110
0	1000	1001
1	1100	1000
1	0101	1010
1	0000	0001

PAGE
DATE



M	A	B	Sum	C4
0	0111	0110	1101	
0				
1				
1				
1				

PAGE
DATE 07/02/23

Computer Instructions:

→ Basic Computer Instruction Format.

1. Memory-Reference Instructions (opcode = 000 ~ 110)

I = 0 → Direct addressing

I = 1 → Indirect addressing.

→ It goes from 8-E (Direct) and 0-6 (Indirect).

Direct: 0xxx, 1xxx ------, 6xxx (Direct memory Instruction)

Indirect: 8xxx, 9xxx ----, Exxx. (I.M.I).

2. Register-Reference Instructions (OP-code = 111, I=0)

→ Any hexadecimal code that starts with 7xxx is a Register-Reference Instructions.

→ 7xxx (Starting)

3. Input-output Instructions: (OP-code = 111, I=1)

→ Any hexadecimal code that starts with Fxxx is a Input-output Instructions.

→ Fxxx (starting).

NOTE:

→ A Basic Computer needs 25 instructions.

→ 3 bits are used as an operational code.

Instruction Set Completeness:

* Functional Instructions:

→ Arithematic, logic & shift Instructions

* Transfer Instruction:

→ LDA, STA → Data transfers between main memory and Processor registers.

PAGE
DATE 08 02 23

* Control Instructions :
→ sequencing and control.

Timing and Control :
Control Unit : Translates from machine instructions to the control signals.
Hard Wired Control :
→ CU is made up of sequential and combinational circuits to generate the control signal.
adv : Fast mode of operations
disad : It requires change of wires.

Instruction Cycle :
→ Fetch - Instruction from memory
→ Decode - the Instruction.
→ Read - Effective address from memory if instruction has indirect address.
→ Execute - the address

Fetch and Decode :
→ After each clock Pulse, SC is incremented By 1
T_0 : AR ← PC
T_1 : IR ← M[AR], PC ← PC + 1 } Fetching
+ Effective address.
T_2 : $D_0, \ldots, D_7$ ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)
↳ Decoding.

⇒ Memory-reference register takes 4 Cycles.
⇒ AR, Register Reference, Input-output register takes 3 Cycles.

PAGE
DATE

* Register-Reference Instructions:

- Identified when D7=1 ; I=0.
- Specified in B0 ~ B11 of IR.
- Execution starts with timing signal T_3.

⇒ $r = D7I'T3$

⇒ $Bi = IR(i), i=0,1,2 \dots 11$

$r = D7I'T_3$ $\quad SC \leftarrow 0$ $\quad$ Clear SC

microoperations like Clear, Compliment	CLA	rB_{11} (7800)	$AC \leftarrow 0$	Clear AC
	CLE	rB_{10}	$E \leftarrow 0$	Clear E
	CMA	rB_9	$AC \leftarrow AC'$	Compliment AC
	CME	rB_8	$E \leftarrow E'$	Compliment E
	CIR	rB_7	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
	CIL	rB_6	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
	INC	rB_5	$AC \leftarrow AC+1$	Increment AC.
Skip of next instruction	SPA	rB_4	if(AC(15)=0) then (PC ← PC+1)	Skip if positive
	SNA	rB_3	if(AC(15)=1) then (PC ← PC+1)	Skip if negative
	SZA	rB_2	if (AC=0) then (PC ← PC+1)	Skip if AC=0
	SZE	rB_1	if (E=0) then (PC ← PC+1)	Skip if E=0
To restore operation, start-stop must be set manually	HLT	rB_0	S ← 0 (S is a start-stop flip-flop)	Halt Computer

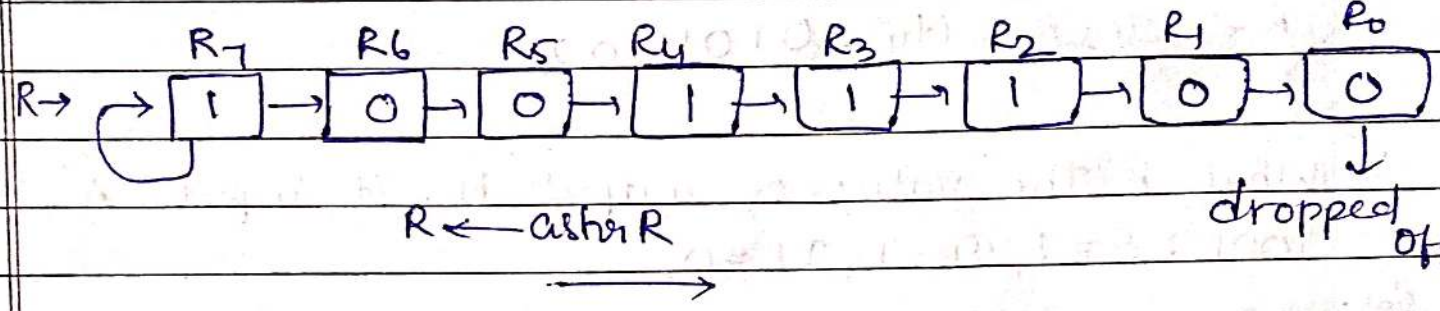
* All these instructions are run in 1 cycle → T_3
* Only one value in the one cycle.
* Subscript remembers which bit is moving
* For skip ; 0 → +ve, 1 → −ve } most significant bit

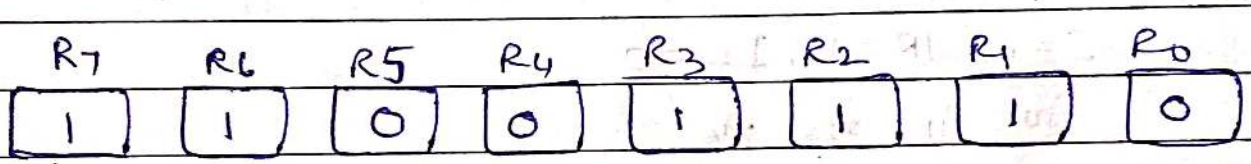
→ Skip – increments the Program Counter.

PAGE
DATE 10 02 23

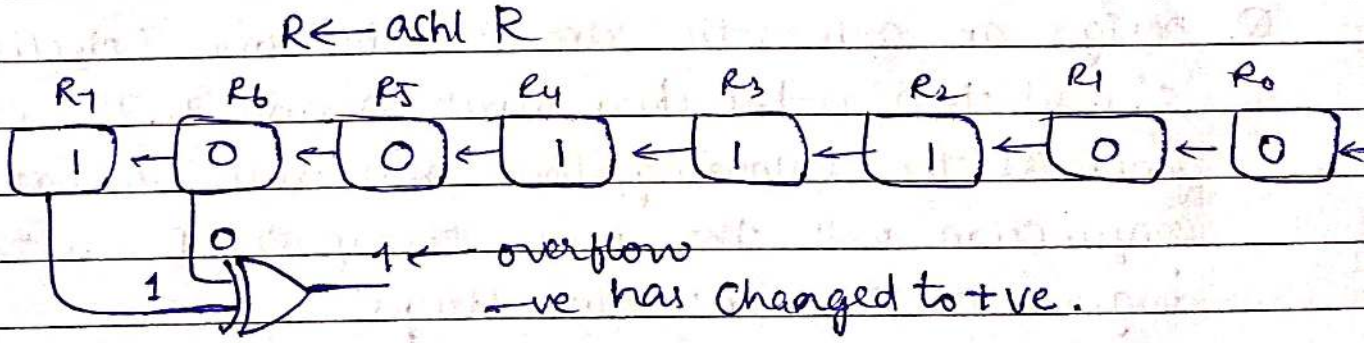
Q. An 8-bit register contains a binary value 10011100. What is the register value after an arithematic shift right? Starting from the initial number 10011100. Determine the register value after the arithematic shift left & state whether there is an overflow.

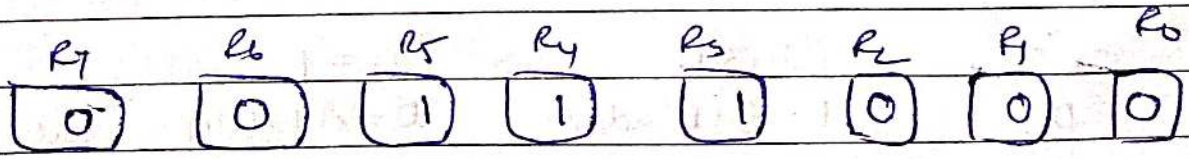
Sol





→ Arithmetic shift will always be on a assigned value.





0110 0011
+ 1
01100100 = +100

0011 0001
+ 1
00110010 = -50

PAGE
DATE

Q. Starting from input value R = 1101 1101. Perform Shl R, cir R, shr R, cil R.

R = 11011101.

Sol: R ← Shl R R(t1) = 10111010

R ← cir R R(t2) = 01011101

R ← shr R R(t3) = 00101110

R ← cirl R R(t4) = 01011100

Q. What is the value of output 'H', if input 'A' is 1001 ; S = 1, IR = 1, IL = 0.

Sol:

S = 1, IR = 1, IL = 0

	A_0	A_1	A_2	A_3
A =	1	0	0	1

	H_0	H_1	H_2	H_3
H =	0	0	1	0

Q. Design an arithmetic circuit with one selection variable 'S' and two n-bit data inputs 'A' and 'B'. The circuit generates the following four arithmetic operations in conjunction with the input carry Cin. Draw the logic diagram for the first two stages.

S	Cin = 0	Cin = 1
0	D = A + B (add)	D = A + 1 (increment)
1	D = A − 1 (decrem)	D = A + B' + 1 (Subtract).

Check the after tutorial.

if M[AR]+1=0 & PC ← PC+1.

→ ISZ D6 M[AR] ← M[AR]+1, J

PAGE
DATE

✡✡ Memory Reference Instructions:

Symbol	operation decoder	symbolic
~~Add~~ AND	D_0	AC ← AC ∧ M[AR]
ADD	D_1	AC ← AC+M[AR], E ← Cout
LDA	D_2	AC ← M[AR]
STA	D_3	M[AR] ← AC
BUN	D_4	PC ← AR
BSA	D_5	M[AR] ← PC, PC ← AR+1

AND to AC:

→ Performs AND logic with AC memory word specified by Effective address (EA).

D_0 AND
- D_0T_4: DR ← M[AR] Read operand [000, 2 cycles]
- D_0T_5: AC ← AC ∧ DR, SC ← 0 and with AC.

D_1 ADD
- D_1T_4: DR ← M[AR] access the memory [001, 2 cycles]
- D_1T_5: AC ← AC+DR, SC ← 0 and E ← Cout.

D_2 LDA
- D_2T_4: DR ← M[AR] read operand [010, 2 cycles]
- D_2T_5: AC ← DR, SC ← 0

D_3 STA
- D_3T_4: M[AR] ← AC, SC ← 0 [011, 1 cycle]
- Accumulator value is transferred to memory add.

∵ Address is read to DR & then to AC because AC is not directly connected to the bus.

PC = 2
DIR

PAGE
DATE 16 02 23

BUN: Branch Unconditionally.

D_4T_4 : PC ← AR, SC ← 0.

LD(PC) = 1

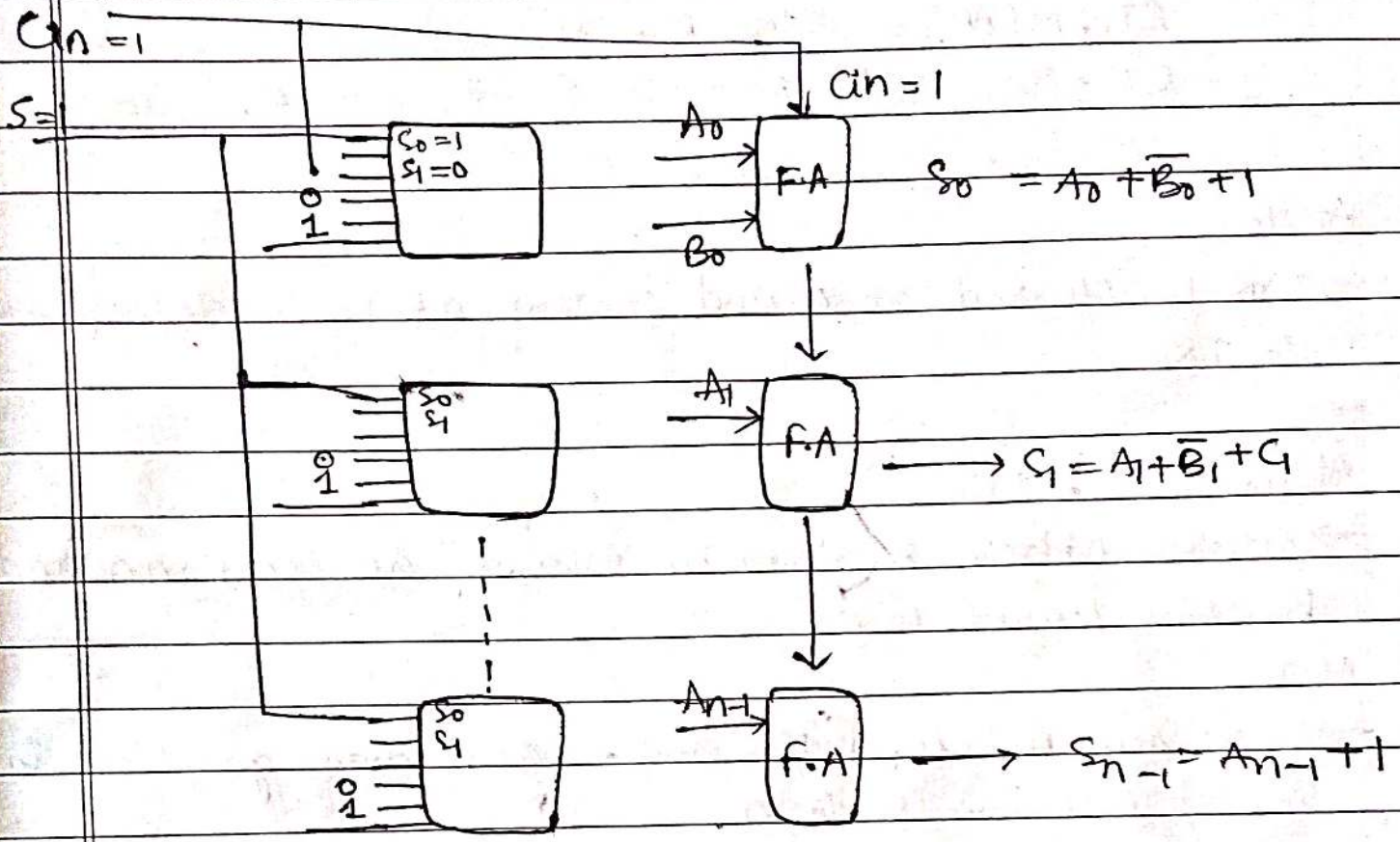
→ giving new address to Program Counter.
→ allows us to specify an instruction out of sequence
Branches unconditionally.

BSA: Branch and Save Return Address:
→ Position of Programs branching.
→ Store address of next instruction in sequence into a memory location specified by effective address.
→ Subroutine - can have multiple instructions.
→ EA+1 transfer to PC serve as 1st Inst. in Subroutine

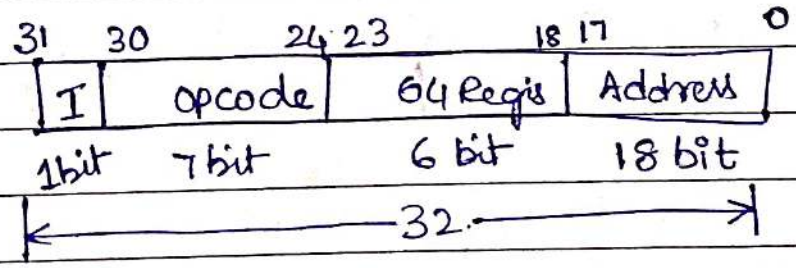
D_5T_4 : M[AR] ← PC, AR ← AR+1
D_5T_5 : PC ← AR, SC ← 0

PAGE
DATE 17/02/23

Q. Check Previous tutorial for Q.



Q. A computer uses a memory unit



i) 7, 6, 18 ii) iii) 32, 18

Q: direct - will have memory effective address
~~Indirect~~ - need 2 reference inst
→ Read inst → read operand

Indirect - 3 reference inst.
- Read inst → Read Effective address → Read operand

PAGE
DATE

Interrupt cycle: (Microoperations)

RT_0: AR ← 0, TR ← PC

RT_1: M[AR] ← TR, PC ← 0

RT_2: PC ← PC+1, IEN ← 0, R ← 0, SC ← 0.

At T_0:

→ AR is cleared at 0 and content of PC is transferred to TR.

At T_1:

→ Return Address is stored in memory at location 0 and PC is cleared to 0.

At T_2:

→ Increment PC clears IEN and R and control goes back to T_0 by clearing SC to '0'.

PAGE
DATE

Interrupt cycle: (Microoperations)

RT_0: AR ← 0, TR ← PC

RT_1: M[AR] ← TR, PC ← 0

RT_2: PC ← PC+1, IEN ← 0, R ← 0, SC ← 0.

At T_0:

→ AR is cleared at 0 and content of PC is transferred to TR.

At T_1:

→ Return Address is stored in memory at location 0 and PC is cleared to 0.

At T_2:

→ Increment PC clears IEN and R and control goes back to T_0 by clearing SC to '0'.

Unit-03

Central Processing Unit (CPU):

PAGE
DATE 28/08/23

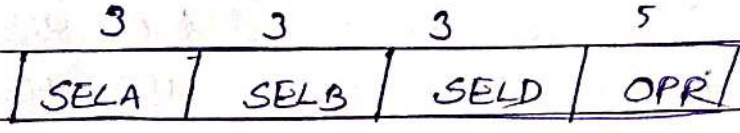
→ Part of computer that Perform bulk of data Processing operations - CPU.

→ Register Set - Store immediate data

→ ALU - Performs required microoperations

→ Control - Supervises the transfer of information.

→ Control word:



⇒ 14 bit code called as Control word.

⇒ SELD = 000, no destination register, but content is available in external output.

ALU Control:

OPR Select	operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subx A-B	SUB
00110	Decrement A	DECA
01000	And A & B	AND
01010	OR A & B	OR
01100	XOR A & B	XOR
01110	Complement A	COMA
10000	Shift Right A	SHRA
11000	Shift left A	SHLA

Sub Micro-operation:

$R_1 \leftarrow R_2 - R_3$

	SELA	SELB	SELD	OPR
field:	R_2	R_3	R_1	SUB
CW:	010	011	001	00101

PAGE
DATE 23/02/23

Symbolic Designation:

Microoperation	SELA	SELB	SELD	OPR	Control word
$R_1 \leftarrow R_2 - R_3$	010	011	001	00101	
$R_4 \leftarrow R_4 \vee R_5$	R_4	R_5	R_4	OR	100 101 100 01010
$R_6 \leftarrow R_6 + 1$	R_6	—	R_6	INCA	110 000 110 00000
$R_7 \leftarrow R_1$	R_1	—	R_7	TSFA	001 000 111 00000
O/P $\leftarrow R_2$	R_2	—	None	TSFA	010 000 000 00000
O/P $\leftarrow$ I/P	—	—	—	TSFA	000 000 000 00000
$R_4 \leftarrow$ shl R_4	R_4	—	R_4	SHLA	100 000 100 11000
$R_5 \leftarrow 0$	R_5	R_5	R_5	XOR	101 101 101 01100

⇒ To clear the Register, we Perform XOR operation with the operand itself.

STACK ORGANIZATIONS:

⇒ Stack - LIFO (Last-In-First-out).

⇒ Its a memory unit with address register.

⇒ Stack Pointer (SP): Register that holds the address for stack.

* PUSH: Insert * POP: Delete.

⇒ Flags: | FULL | EMPTY |

→ when stack is full; FULL = 1

→ when stack is Empty; EMTY = 1.

PUSH:

$SP \leftarrow SP + 1$	increment SP
$M[SP] \leftarrow DR$	write item on top of stack

If (SP = 0) Then (FULL ← 1) Check if stack is full
EMTY ← 0 Mark Stack not Empty

PAGE
DATE

POP:

DR ← M[SP] Read item from top of stack
SP ← SP − 1 Decrement SP
If (stack = 0) then (Empty ← 1) Check if stack is Empty
Full ← 0 Mark the stack not full.

Memory stack organization:

→ Computer memory partitioned into three segments
* Program (PC) - Fetch to read instructions
* Data (AR) - Execution to read operand
* Stack Segments (SP) - Push and pop items
→ For checking stack overflow - Software is needed
→ Upper limit - to be checked after push operation
→ Lower limit - to be checked after pop operation.

Tutorial
24-02-23

(i) IR ← M[PC] ⇒ AR ← PC
IR ← M[AR]

(ii) AC ← AC + TR
DR ← TR
AC ← AC + DR

(iii) DR ← DR + AC
⇒ AC ← DR + AC
DR ← AC

PAGE
DATE

(i) 0001 0000 0010 0100
$(1\ 0\ 2\ 4)_{16}$ ⇒ ADD (024) / ADD M[024]

(ii) 1011 0001 0010 0100
$(B\ 1\ 2\ 4)_{16}$ ⇒ STA I (124) / STA M[M[124]]

(iii) 0111 0000 0010 0000
$(7\ 0\ 2\ 0)_{16}$ ⇒ INC

Q. The content of AC in the basic Computer is Hexadecimal A937 at the initial value of E is 1. Determine the contents of AC.....

08-03-23

Sol:
AC = A937
E = 1 ; PC = 021

	AC	E	PC	AR	IR
CLA	0000	1	022	800	7800
CLE	A937	0	022	400	7400
CMA	56C8	1	022	200	7200
CME	A937	0	022	100	7100
CIR	D49B	1	022	080	7080
CIL	526F	1	022	040	7040
INC	A938	1	022	020	7020
SPA	A937	1	022	010	7010
SPN	A937	1	023	008	7008
SZA	A937	1	023	004	7004
SZE	A937	1	022	002	7002
HLT	—	—	—	—	—

56C8

CIR:
1010 1001 0011 0111
1101 0100 1001 1011
D 4 9 B
0101 0010 0110 1111
5 2 6 F

PAGE
DATE 28/02/23

Reverse Polish Notation:

→ Infix Notation - each operator written between operands.

Eg:

A+B - Infix

+AB - Prefix / Polish Notation

AB+ - Postfix / Reverse Polish Notation [RPN]
useful for stack Manupulation.

Instruction Format:

→ Instruction fields < op-code field - operation to be performed
Mode field - how address field is to be interpreted. (effective address).

* Single Register accumulator:

→ Operation performed with an implied accumulator register and address field.

* General register organisation:

→ Generally requires three register address field.

Eg: ADD R_1, R_2, R_3

$R_1 \leftarrow R_2 + R_3$

→ Very fast and have long registers and long addresses

ADD R_1, R_2

$R_1 \leftarrow R_1 + R_2$.

* Processor Organization: Pop and push instruction with address field.

* Register address is smaller than the memory address.

PAGE
DATE 01/03/23

Single Accumulator Orga:

Eg: Add X.

AC ← AC + M[X] (or) AC ← AC + R[X].

Three Address Instructions:

→ Each address field for Processor Register or Memory operand.

X = (A+B) * (C+D).

(i) ADD R_1, A, B $R_1 \leftarrow M[A] + M[B]$

(ii) ADD R_2, C, D $R_2 \leftarrow M[C] + M[D]$

(iii) Multiply X, R_1, R_2 $M[X] \leftarrow R_1 * R_2$

(or)

load R_1	Move R_1, A.	$R_1 \leftarrow M[A]$
	Add R_1, B	$R_1 \leftarrow R_1 + M[B]$
	Mov R_2, C	$R_2 \leftarrow M[C]$
	Add R_2, D	$R_2 \leftarrow R_2 + M[D]$
	MUL R_1, R_2	$R_1 \leftarrow R_1 * R_2$
	Mov X, R_1	$M[X] \leftarrow R_1$

One address Instructions:

→ used as Implied accumulator (AC)

Eg: X = (A+B) * (C+D)

load A	AC ← M[A]
Add B	AC ← AC + M[B]
Store T	M[T] ← AC
load C	AC ← M[C]
Add D	AC ← AC + M[D]
MUL T	AC ← AC * M[T]
Store X	M[X] ← AC

PAGE
DATE

Zero Address Instructions: Push, Pop and RPN

Eg: $X = (A+B) * (C+D)$.

Push A	TOS ← A
Push B	TOS ← B
ADD	TOS ← (A+B)
Push C	TOS ← C
Push D	TOS ← D
ADD	TOS ← (C+D)
MUL	TOS ← (C+D) * (A+B)
Pop X	M[X] ← TOS ⇒ 'X' is not an address, it's an operand

* Super Computers are made using the stack organization.

03-22

☆ Addressing Modes:

→ Modifying the address field of the instruction.

* Implied Mode:

→ address of the operands are specified implicitly in the definition of the instruction.

→ No need to specify address in the instruction

→ EA = AC or EA = Stack [SP]

→ Eg of basic computer - CLA, CME, INR.

→ Address is given within the instruction.

* Immediate Mode:

→ Instead of specifying the address of operand, Operand itself is specified.

→ Sometimes require more bits than the address.

→ fast to acquire an operand

* XR → Index register

PAGE
DATE

* **Register Mode:**
→ Address specified in the operand instruction is the register address.
→ Shorter address than your memory address

* **Register Indirect mode:**
→ Instruction specifies a register which contains the memory address of the operand.
→ EA = [IR(R)] [[X]: Content of X].

* **Autoincrement or Autodecrement Mode:**
→ Increment and decrement happens in register value.
* Autoincrement ⇒ $R_1 \leftarrow 13$ execute
$R_1 \leftarrow R_1 + 1$
* Decrement ⇒ $R_1 \leftarrow R_1 - 1$ then execution.

* **Direct Address:**
→ Instruction Specifies memory address which can be used directly to access the memory.
* **Indirect Address:**

* * **Relative Addressing Mode:**
→ The address fields of an instruction specifies the Part of the address, which can be used along with a designated register to calculate the address of the operand.
→ Address field of the instruction is short.
→ Large physical memory can be accessed with a small no. of address bits.

PAGE
DATE

Q. An instruction at address 021 in the basic computer has I=0, an operation code of the AND instruction and an address part equal to 083 (HD). The memory word at address 083 containing and operand B8F2 and content of AC is A937.

Sol: I=0 ; AC=A937 ; I=0 (direct) ; AND - $(083)_{16}$.

	PC	AR	DR	AC	IR				
Initial	021			A937		1010	1001	0011	0111
AND	022	083	B8F2	A832	0083	and 1011	1000	1111	0010
ADD	022	083	B8F2	6229	1083	1010	1000	0011	0010
						A	8	3	2

Date / /
Page No. 07/07/23

Data Transfer Instructions:

@ - Indirect
$ - Relative
# - Immediate operand
( ) - Index addressing
(R) - Register indirect.

Data Manupulation:

→ Performs arithmetic, logic and shift operations

NOTE:

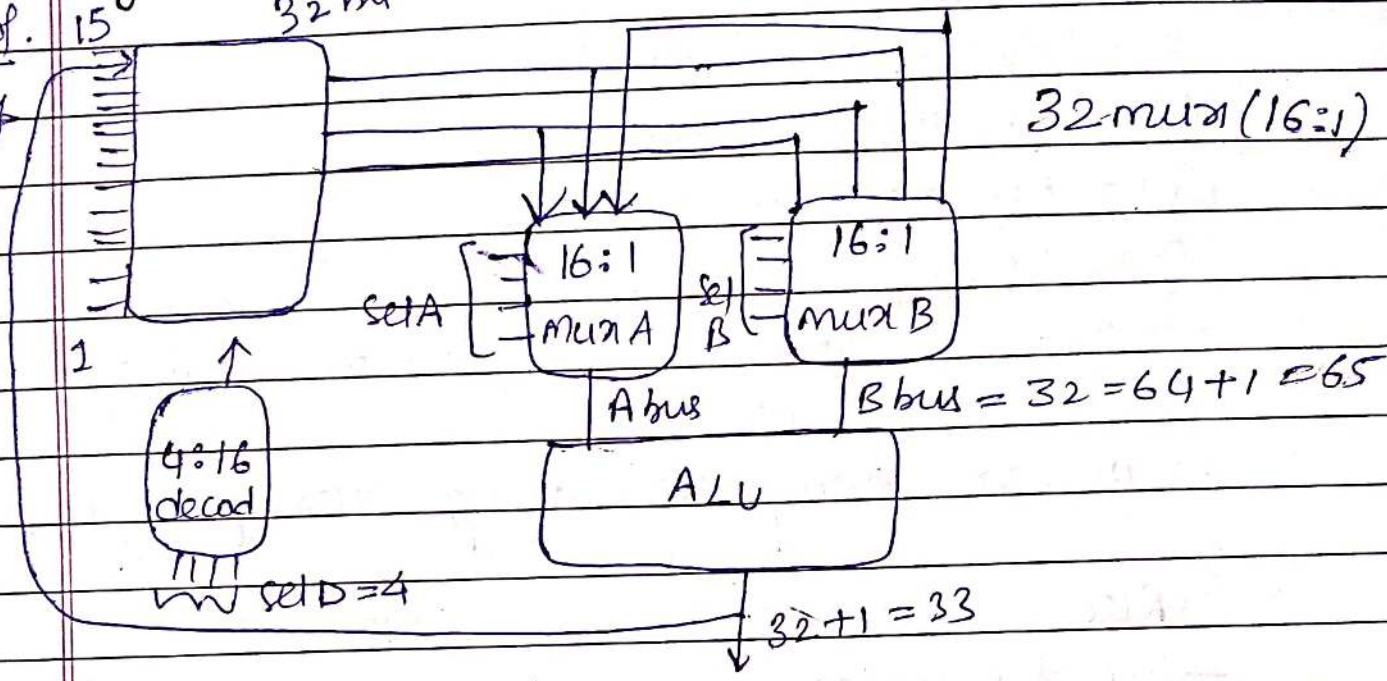
→ we can compare using XOR gate as well

RISC	CISC
→ less instruction	→ more instructions
→ fixed length inst	→ variable length inst
→ less addressing mode	→ more addressing mode
→ Easy to decode	→ Difficult to decode.
→ Software / Hardware instruction	→ Microprocessor are required.
→ single cycle	→ Multiple cycles.

Date 07/03/23
Page No.

Tutorial * CPU based on register set organization has 16 registers and 32 bit in each.

Sol. 15 32 bit



SELA	SELB	SELD	OPR
4	4	4	6-bit

= 18 bit Control word.

Q. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A Processor register R_1 contains the number 200.

Memory

300	Inst (opcode)
301	Address = 400
302	Next Ins

Sol:

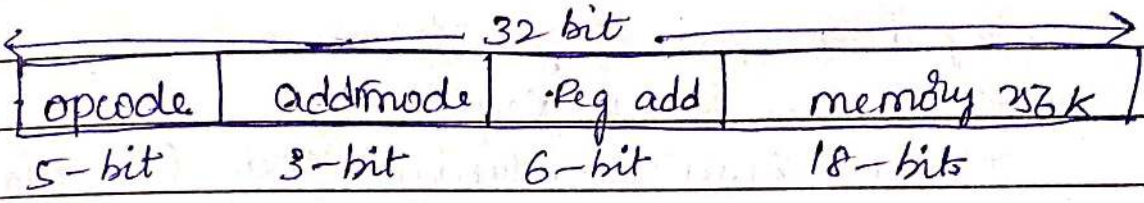
1. Direct = 400
2. Immediate = 301
3. Relative = 302 + 400 = 702
4. Register } = 200
   Indirect }
5. Index with R_1 as the index register = 200 + 400 = 600.

Q. Convert the following numerical arithemetic expression into R.

Ans: $(3+4)[10*(2+6)+8]$

$\rightarrow 34+26+10*8+*=616.$

Q. The memory unit of a computer has 256K words of 32 each. The computer has an instruction format with four fields.



UNIT-04

Date 23/03/23
Page No.

INPUT AND OUTPUT ORGANISATION

→ Peripherial Devices – I/p & O/p devices. to computer.
USB – universal Serial Bus.
→ Devices that are under Direct control of a Computer are said – Connected on-line.
*Three types of Pheripheral
→ Input pheripheral → output Phesripheral
→ Input – output Pheriphusal.
CRT – cathode Ray tube.

Input – output organisation :

→ It's an ASCII key.
→ ASCII (American standard code for Information Interchange)
→ 7 bit code (2^7=128 characters can be included)
→ 94 are printable & 34 non-Printable
↳ alphabets, numbers
↳ Space, tab, backspace, delete.
↳ Control Characters.
↳ information seperator etc.,
→ 7 bit ⇒ 00–7F (0–127)

→ Video monitors are the most commonly used Peripherals.
→ They consists of a keyboard as the input device and a display unit as the output device.

Date 24/03/23
Page No.

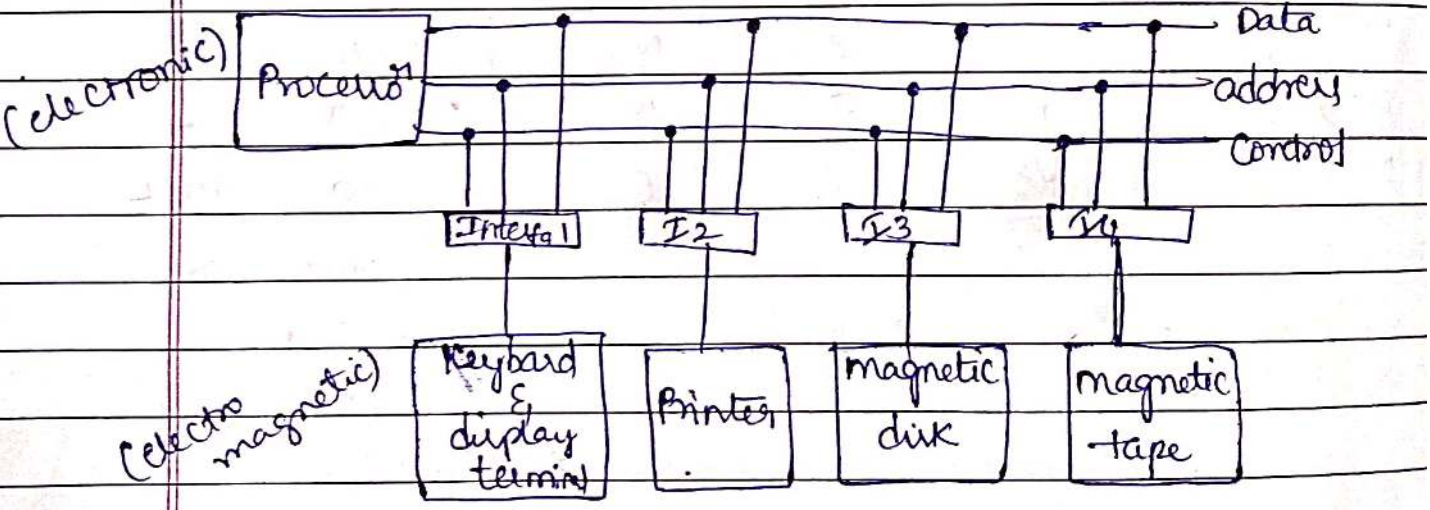
ASCII CODE:

For CSE,

Binary - 0100 0011 0101 0011 0100 0101
Hexa - 4 3 5 3 4 5
Decimal - 67 83 69

Control Characters: 34 non-printable Characters.
* Format Effectors - Control layout of Printing
* Information Seperator - Seperate data into divisions
* Communication Control characters.
  ↳ These are useful during the transmission of text b/w remote terminals.
  Eg: Start of text ; End of text

Input - Output Bus and Interface:



Interface:

→ Synchronizes the dataflows and supervises.
→ Interface does - transfer rate b/w peripheral & cpu or memory (data formats)

Date 24/03/23
Page No.

→ Provides Signals for the Peripheral controller
→ Decodes the device address (device code)
→ Decodes the Commands (operation).

* Conversion of Signals required
→ Peripheral – Electromechanical & magnetic device
→ CPU or memory – Electronic device
* Data transfer Rate
→ Peripheral – usally slower * operational differ
→ CPU/memory – faster
* Data formats
→ Peripheral – Bits, Bytes, Blocks
→ CPU/memory – word

I/O Bus and Interface:
* Control Command – activate Peripheral device & inform what to do
* Status Command – Test various Status Condition in the interface & Peripherals.
* Data O/p Command – Causes the interface to respond by transferring data from bus to one of its registers.
* Data I/p Command – Recieves an item from data from Peripheral & places in its buffer register.

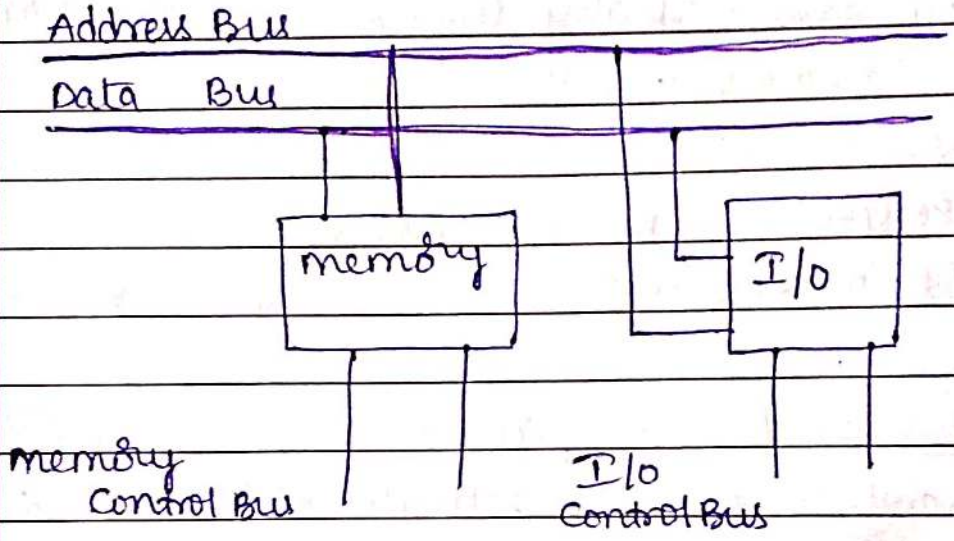
Date / /
Page No.

Function Of Buses:

* Memory Bus is for information transfers b/w CPU and the memory.
* I/O Bus is for information transfers between CPU and I/O devices through their I/O interface.

IOP- Input output Processor

Isolated I/O:



advantage:

→ full memory space is available as I/O is isolated

Disadvantage:

→ special Instruction * Difficult for Programming.

* Address for I/O here is called Ports (Port mapped I/O) also known as (I/O (Input) mapped I/O (Input output)).
* Different read write Instruction for both I/O and memory. → for I/O → IN/out
  memory → load / Mov / Push or pop.

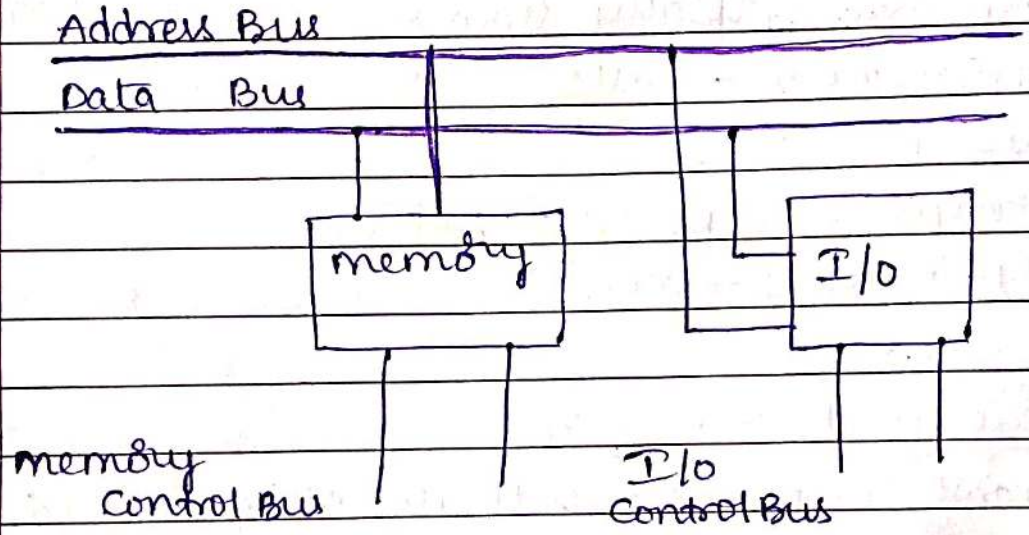
Date / /
Page No.

Function Of Buses:

* Memory Bus is for information transfers b/w CPU and the memory.
* I/O Bus is for information transfers between CPU and I/O devices through their I/O interface.

IOP- Input output Processor

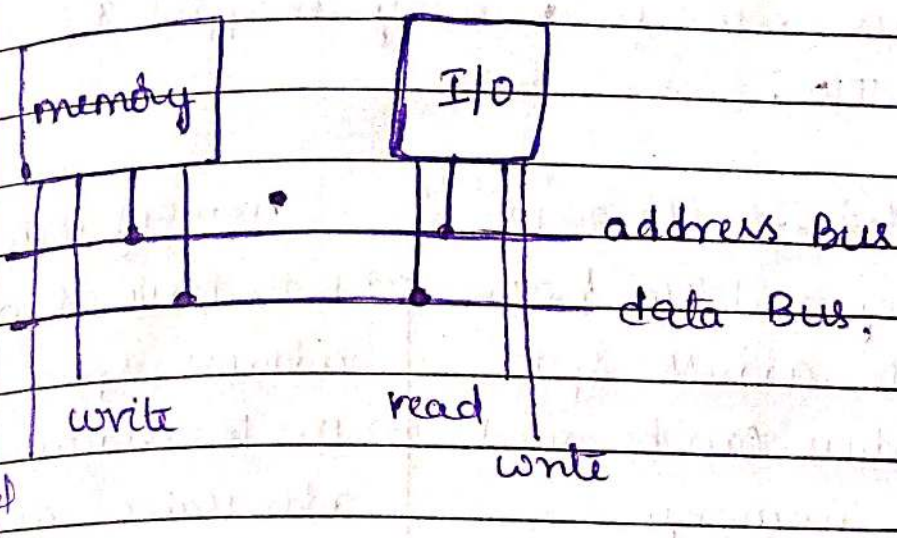
Isolated I/O:



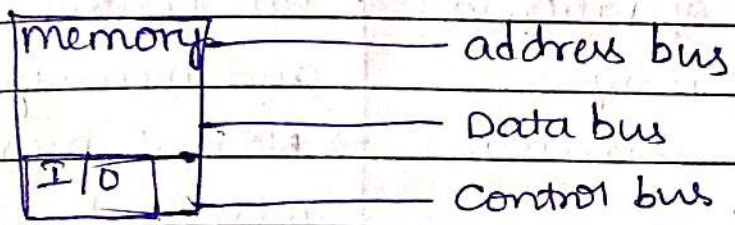
advantage:
→ full memory space is available as I/O is isolated

disadvantage:
→ special Instruction * Difficult for Programming.

* Address for I/O here is called Ports (Port mapped I/O also known as (I/O mapped I/O).
* Different read write Instruction for both I/O and memory. → for I/O → IN/out
memory → load / Mov / Push or pop.



Memory Mapped I/O :



* Common bus (data & address) and control lines are common for I/O and memory.
* advantage - control lines are same & reduced.
disadvantage - memory is reduced
* Single set of instructions of read/write.

I/O versus Memory Bus :

3 ways that computer buses can be used to communicate with memory and I/O :

1. Use two seperate buses, one for memory & I/O.
2. Common bus for both memory & I/O but have seperate control lines for each.
3. use one common bus for memory and I/O with common control lines.

* Differences between memory mapped I/O and Isolated I/O.

Isolated I/O (I/p output mapped)	memory mapped I/O
→ memory and I/O have seperate address space.	→ Both have same address space.
→ All address can be used by the memory	→ Due to addition of I/O addressable memory becomes less for memory.
→ Seperate Instruction Control Read & Write operation in I/O and memory	→ Same Instruction can Control both I/O and memory
→ In this I/O address are called Ports.	→ Normal memory address for both.
→ More efficient due to seperate buses.	→ Lesser Efficient
→ larger in size due to more buses.	→ Smaller in size.
→ Complex due to seperate logic is used to control both	→ Simpler logic is used as I/O is also treated as memory only.

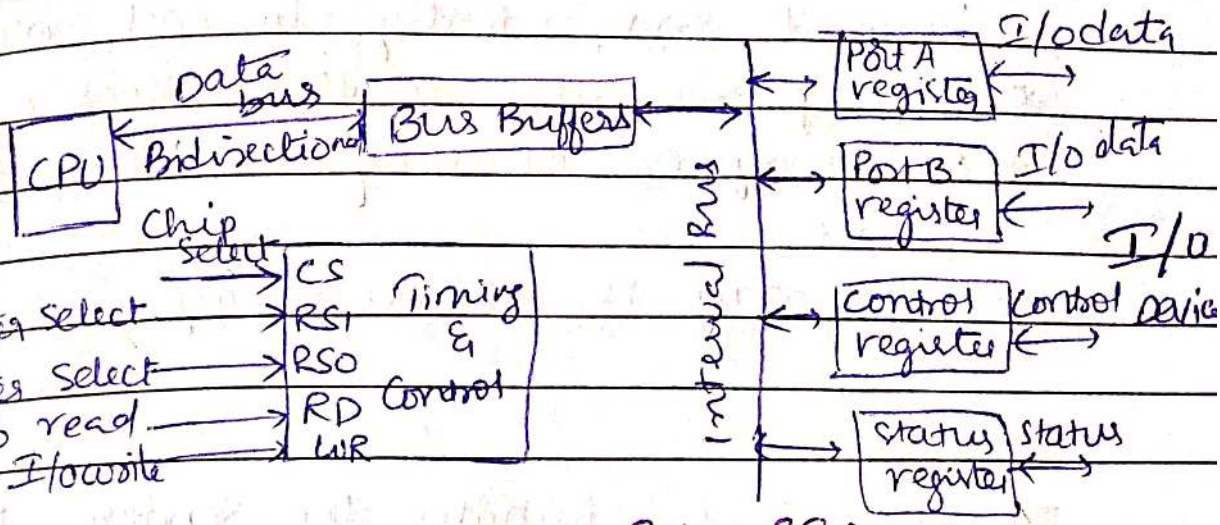
Date / /
Page No.

* Differences between memory mapped I/O and Isolated I/O.

Isolated I/O (I/P output mapped)	memory mapped I/O
→ memory and I/O have seperate address space.	→ Both have same address space.
→ All address can be used by the memory	→ Due to addition of I/O addressable memory becomes less for memory.
→ Seperate instruction control read & write operation in I/O and memory	→ Same Instruction can Control both I/O and memory
→ In this I/O address are called Ports.	→ Normal memory address for both.
→ More Efficient due to seperate buses.	→ Lesser Efficient
→ larger in size due to more buses.	→ Smaller in size.
→ Complex due to seperate logic is used to control both	→ Simpler logic is used as I/O is also treated as memory only.

Date / /
Page No.

Example of I/O Interface:



Register Selected	CS	RS1	RS0	
None – bus in high impedence	0	X	X	
Port-A register	1	0	0	
Port-B register	1	0	1	
Control register	1	1	0	— Case 1
Status register	1	1	1	— Case 2

* Asynchronous Data Transfer:

→ Synchronous data transfer: Pulses are applied to all the registers within a unit and all data transfer among internal registers occurs simultaneously during the occurrence of a clock pulse.

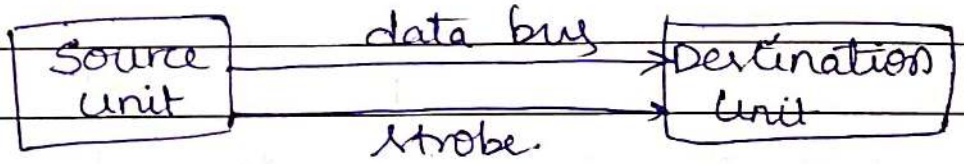
* If the Devices have different clock pulse, then that's called a Asynchronous data transfer.

Date / /
Page No.

* Two units such as CPU and I/O are designed independently to each other.
* It requires an another clock pulse.
* One-way of achieving this is by means of STROBE
* Other way is Hand Shaking

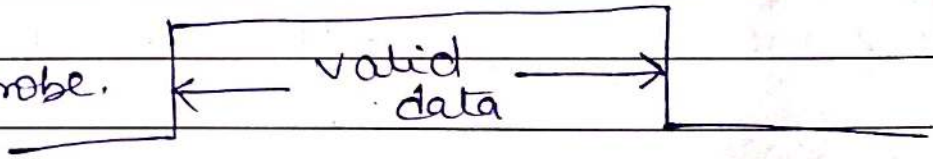
STROBE :

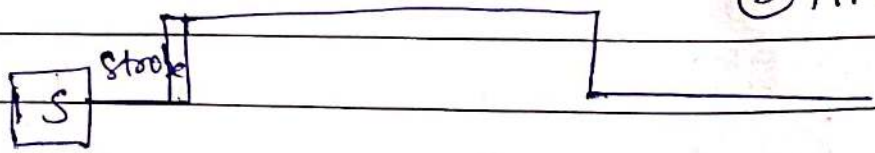
→ After strobe initiates the sender, the data transfers between sender & reciever.



@ block diagram.

fig: Source initiated strobe.



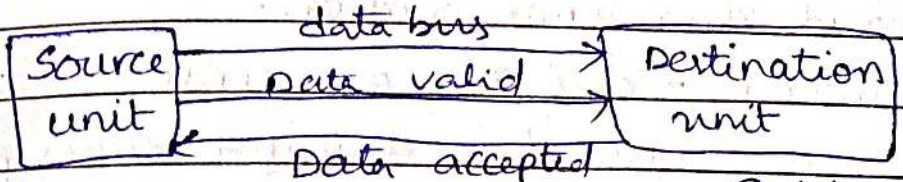


(b) Timing signal.

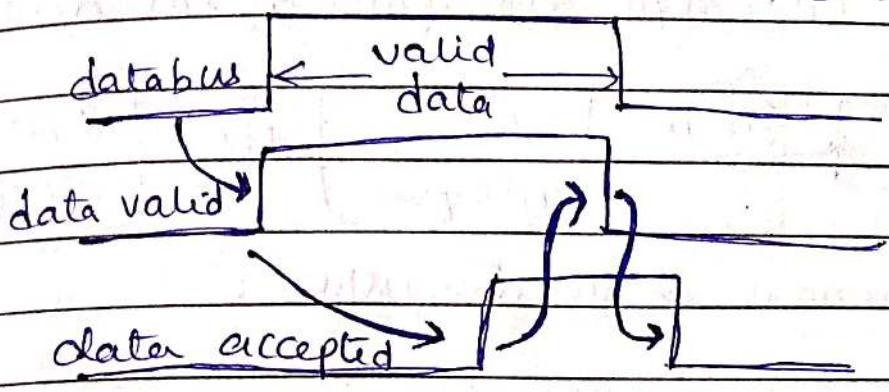
disadvantages : NO acknowledgement.

Date / /
Page No.

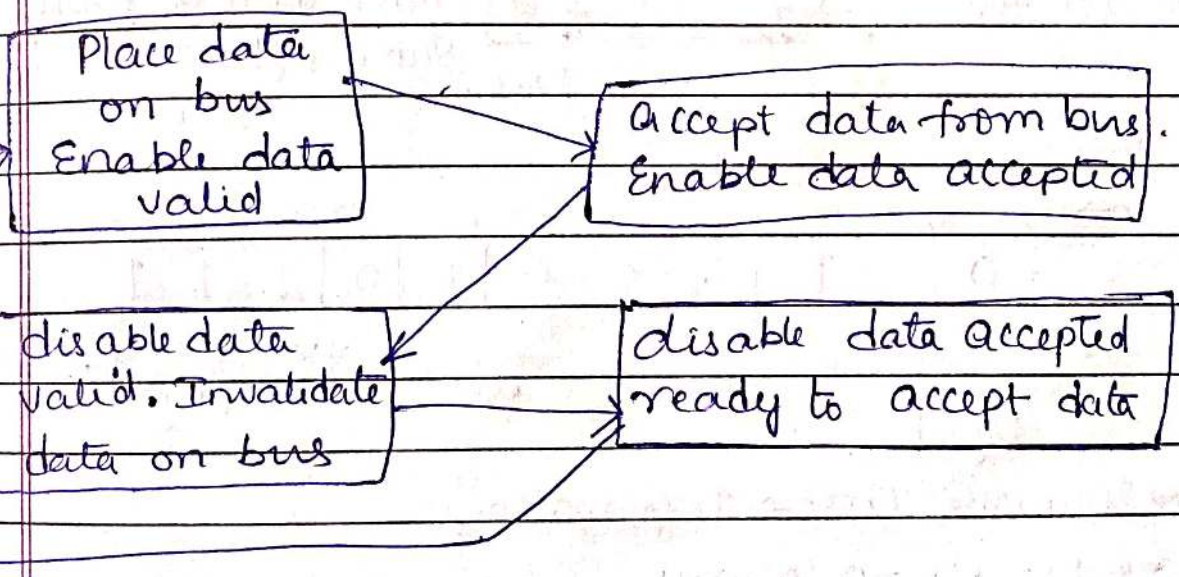
Source initiated using Handshaking :



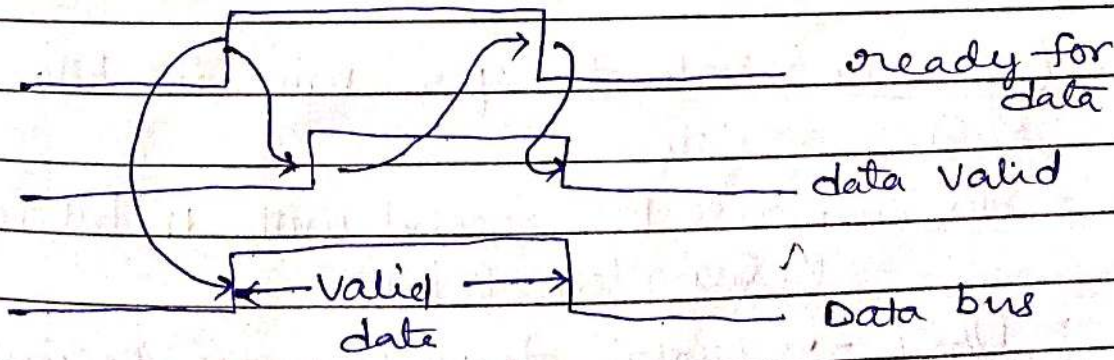
@ Block diagram



(b) Timing Signal.



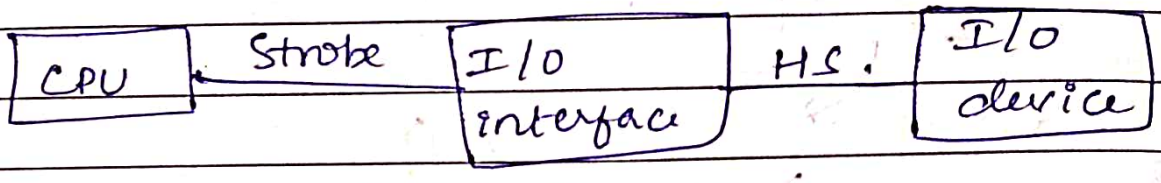
Destination Source : HS :-



for Timing Signal

Date 29/03/23
Page No.

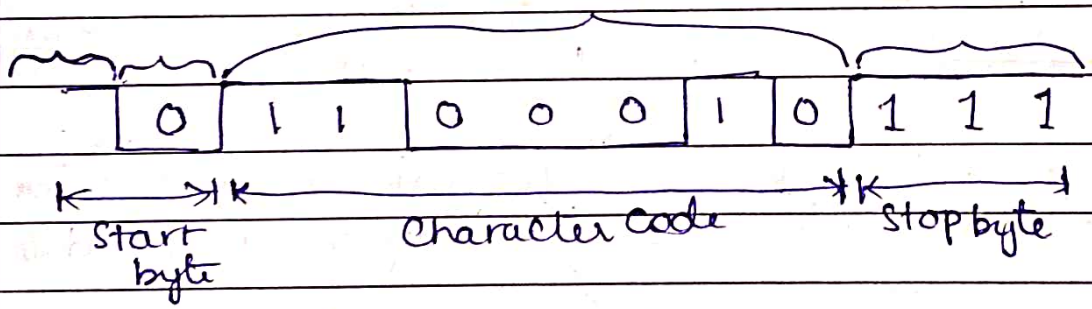
Advantage: HS
→ Valid data confirmation.
→ Timeout - Error in one unit is faulty, & no data transfer takes place.
→ used b/w I/O Interface & I/O device.



Asynchronous Serial Transfer:

Serial: S → 1100011101 → D If data is 8 bits.
8 bits → 8 sec (takes more time)

Parallel: S ⇉ D For a data of 8 bits, 8 lines should be there b/w S & D.
8 bits 1 sec. (takes less time)

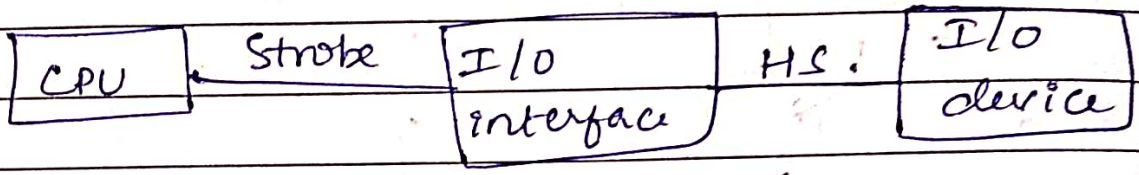


→ Character Code ⇒ Information
→ Start bit = 0 always.
→ Stop bit = 1 always

Baud Rate: Data transfer rate in bits per second.
↳ group of bits.
→ 10 Character per second with 11 bit format = 110 bit per s
(1 Character = 11 bit)
→ UART - universal Asynchronous Reciever - Transmitter

Date 29/03/23
Page No.

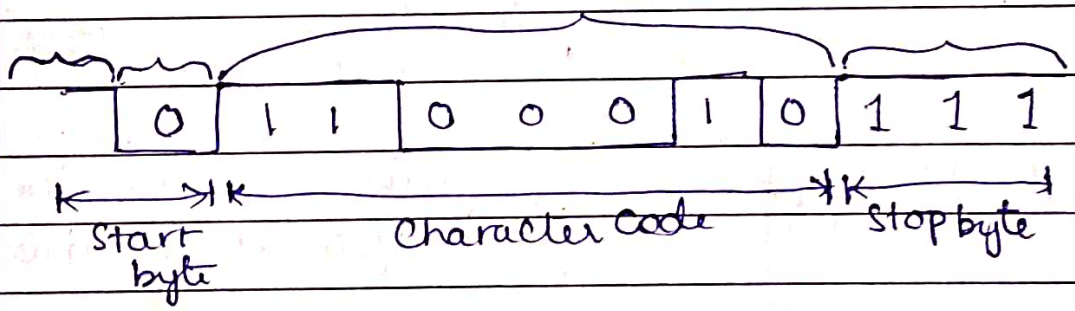
Advantage: HS
→ Valid data confirmation.
→ Timeout - Error in one unit is faulty, & no data transfer takes place.
→ used b/w I/O Interface & I/O device.



Asynchronous Serial Transfer:

Serial: S → 1100011101 → D If data is 8 bits.
8 bits → 8 sec (takes more time)

Parallel: S ⟹ D For a data of 8 bits, 8 lines should be there b/w S & D.
8 bits 1 sec. (takes less time)



→ Character Code ⇒ Information
→ Start bit = 0 always.
→ Stop bit = 1 always

Baud Rate: Data transfer rate in bits per second.
↳ group of bits.
→ 10 Character per second with 11 bit format = 110 bit/s
(1 Character = 11 bit)
→ UART - universal Asynchronous Receiver - Transmitter

Date / /
Page No.

ACI – Asynchronous Communication Interface.

→ Transmitter register –
→ accept a data byte (from CPU) through the data bus
→ transferred to a shift register for serial trans.

Recieves Register –
→ Recieves serial info into another Shift registe
→ Complete data byte is sent to the recieves register.

Status register bits :
→ used for I/o flags and for recording errors (Parity error, framing error).

Control register bits : read and write
→ Define baud rate, no. of bits in each Character, whether to check Parity & no. of Stop bits.

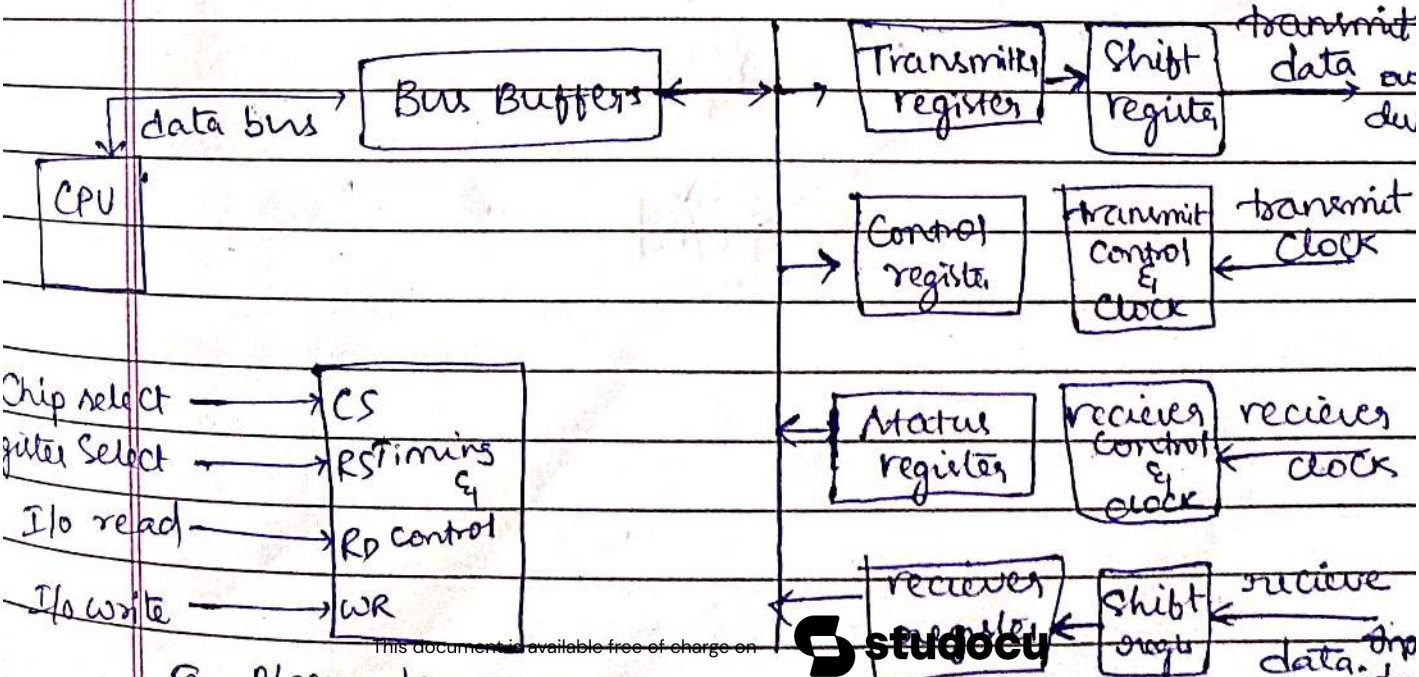


fig: Block diagram of a typical asynchronous

Date 29/03/23
Page No.

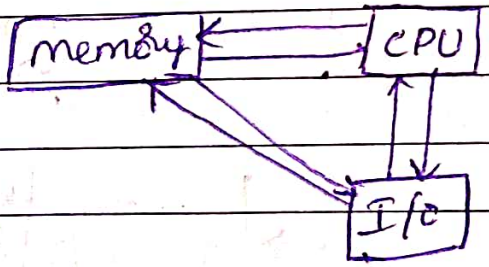
CS	RS	oper	register selected
0	x	x	none
1	0	WR	Transmitter reg
1	1	WR	Control reg
1	0	RD	receiver reg
1	1	RD	Status reg

Modes of Transfer:

→ Data transfer to or from Peripheral can be handled in one of 3 Possible modes:
- → Programmed I/O
- → Interrupt-initiated I/O
- → Direct Memory acces (DMA)

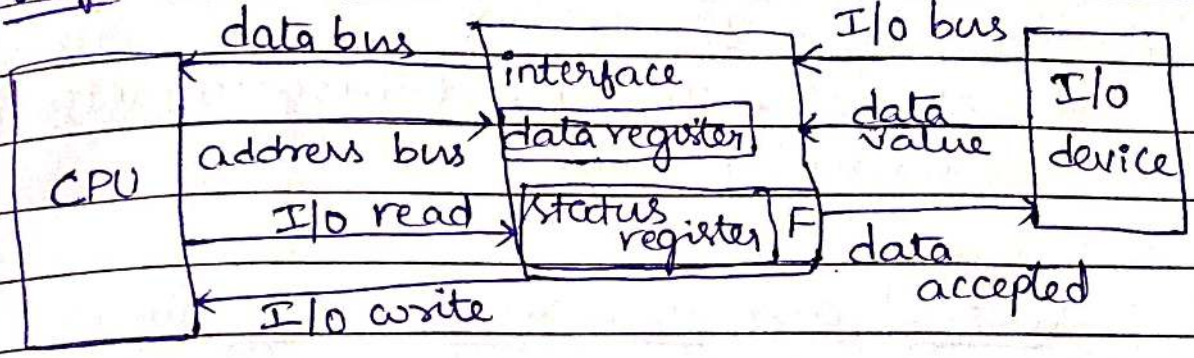
→ Communication b/w I/O and memory is done through CPU.

→ The received Binary info from an external device is stored in Memory.



Date 30/03/23
Page No.

Programmed I/O :



→ If flag = 1 - CPU reads the data from the data register.
→ to and from a CPU register to Peripheral - transfer of data.
→ disadv - It is a time consuming Process since it keeps the Processor busy needlessly
→ It can be avoided by using interrupt.

Interrupts :

→ It is a request from H/W device or S/W Program for immediate Service by Processor.

Adv: Efficiency of CPU is improved.
disad: overhead required to service the interrupt
→ save status to Processor & restore it.

PSW - Program Status word :

S	V	FG1	FG0	C	Z

Date / /
Page No.

Types of Interrupt :

* External * Internal (Hardware Interrupt)
* Software Interrupt

Vectored and non-vectored Interrupt :

* Non-vectored :
→ Branch address is assigned to a fixed location in memory.
→ Device doesn't Provide the address of ISR.

* Vectored :
→ Source that interrupts, supplies the branch information to the computer – interrupt vector.
→ Modify used when CPU has multiple Peripherals Connected to the System bus.

Interrupt Initiated I/O :

→ CPU Proceeds to execute another Programs and interface keeps monitoring the flag.
→ When CPU is ready to transfer – generate interrupts.

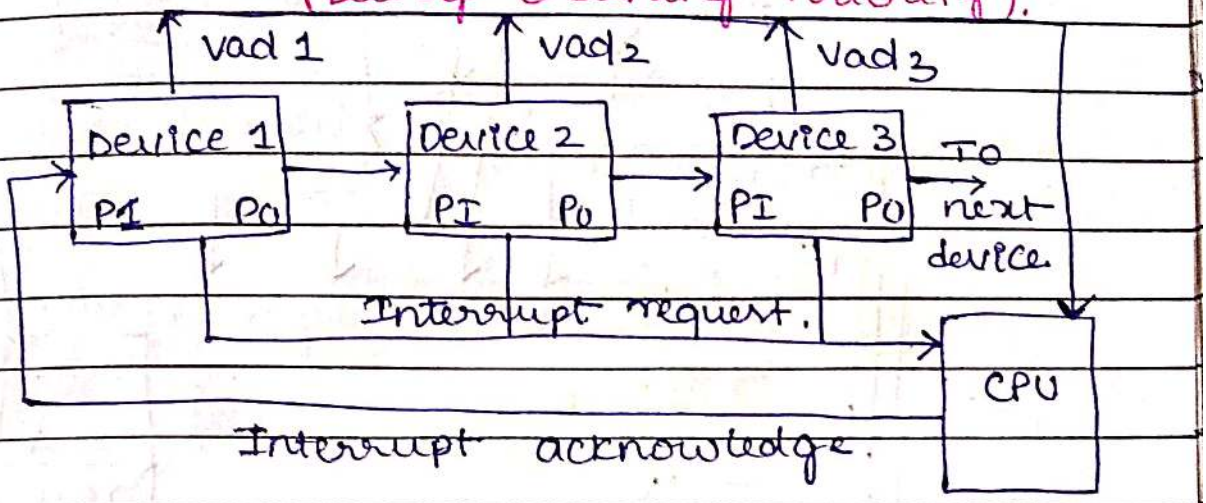
Daisy Chain Priority Interrupt :

→ System that establish Priority, to determine which Condition to be serviced first when multiple interrupt requests arrive simultaneously.

Date 04/04/25
Page No.

two method for PI:
→ Software - Polling
→ Hardware - Serial or Parallel Connection.

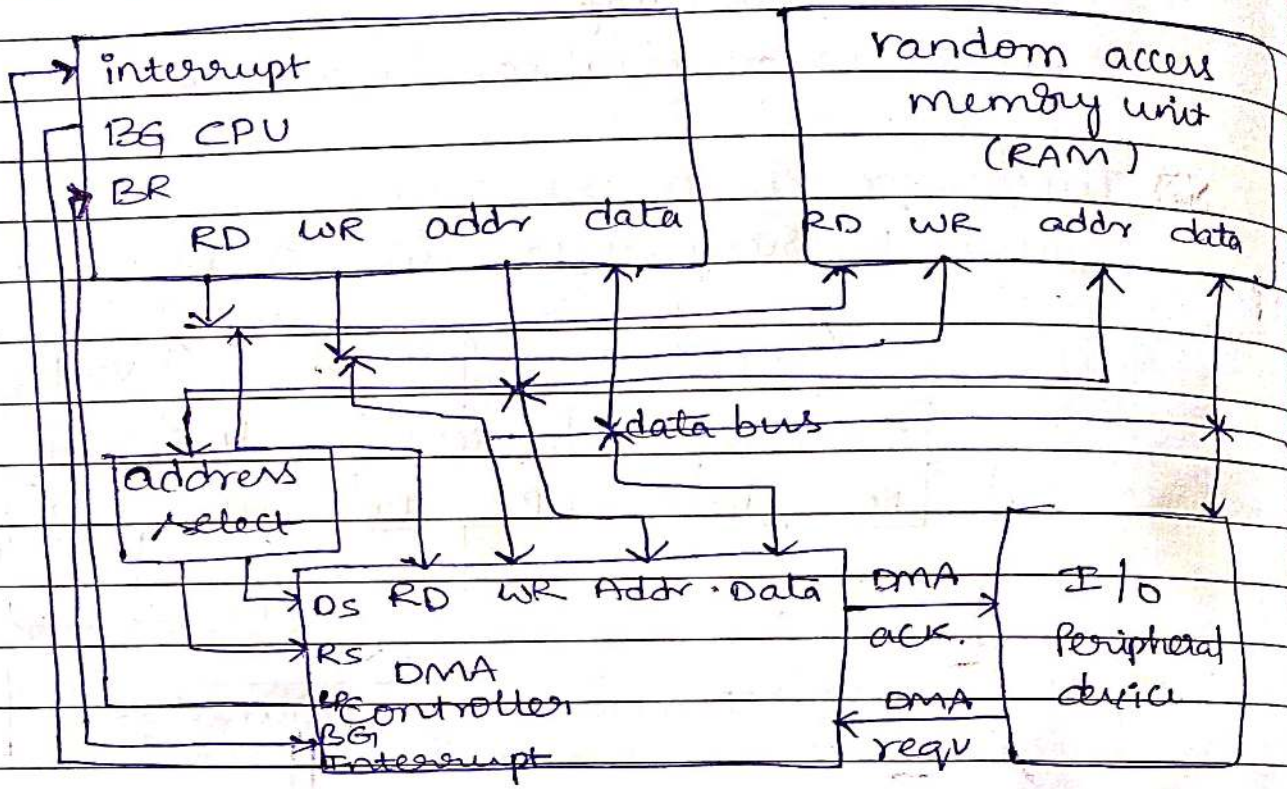
** Interrupt Initiated I/O:
Serial Priority Interrupt Connection:
(Daisy-Chaining Priority).



Direct Memory Access (DMA):
→ allows I/O devices to transfer data exactly to/from memory without CPU intervention.
→ DMA Controller - Interface which takes over the buses to manage the transfer directly b/w memory & I/O device, freeing CPU for other tasks.
→ Address register: Contains an address to specify desired location in memo
→ Word Count: Holds no. of words to be transferred.
→ Control register: will have the operations to be performed.

Date / /
Page No.

DMA Transfer:



BR – Bus request → RD/WR
BG – Bus grant → Bit of data.

If BG → 0 then CPU can communicate with DMA register.

If BG → 1 CPU left buses & DMA can Communi-cate directly with the memory.

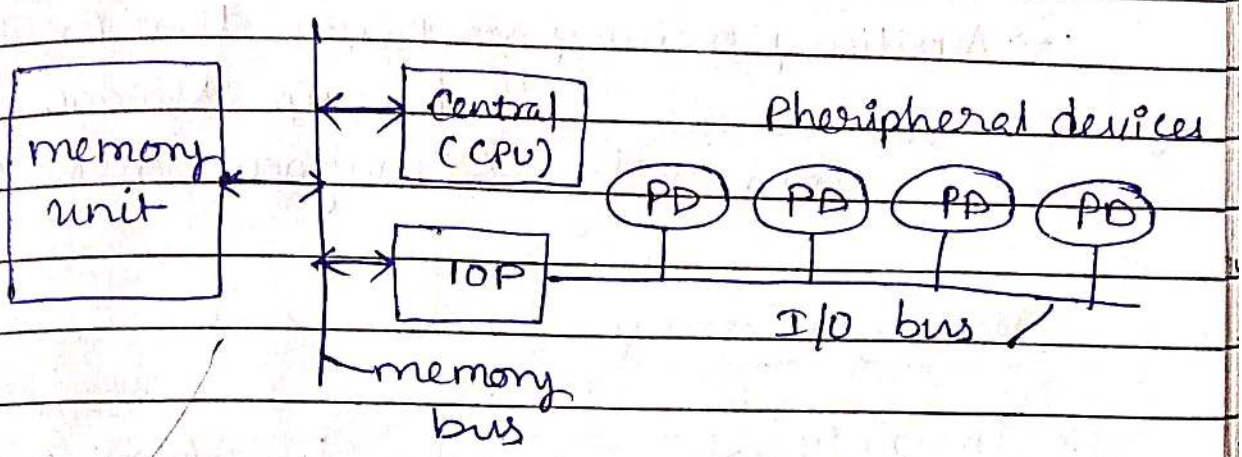
DMA transfer can be made in several ways.

* Burst transfer – memory word is transferred in continuous burst.
* Cycle Stealing – allows DMA Controller to transfer one data word at a time.

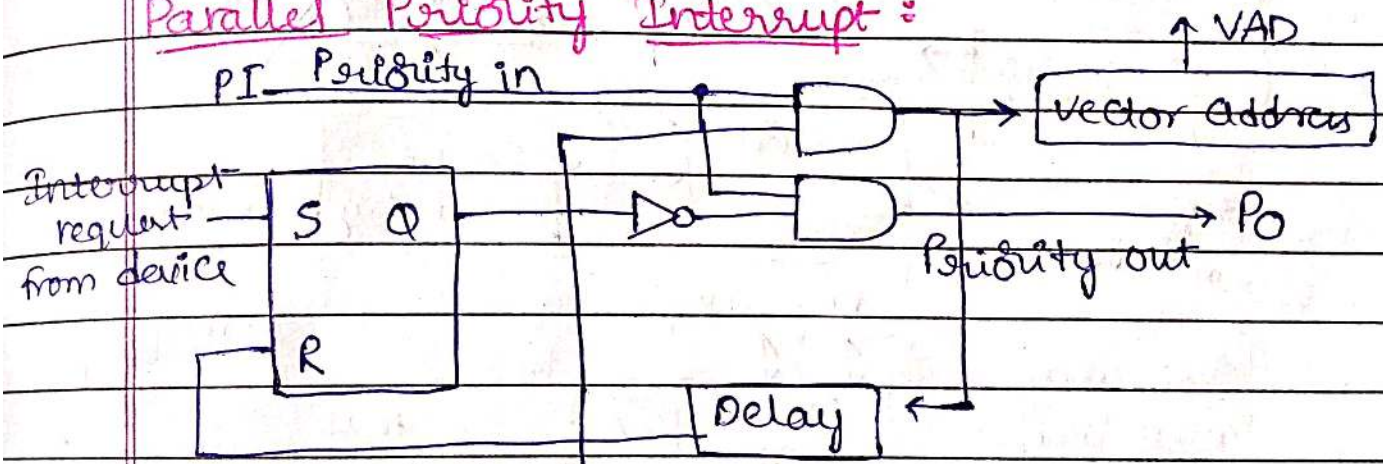
Date / /
Page No.

I/O Processor (IOP):

→ designed to handle I/O Processing.
→ unlike DMA controller, IOP can fetch and execute its own instruction.



Parallel Priority Interrupt:



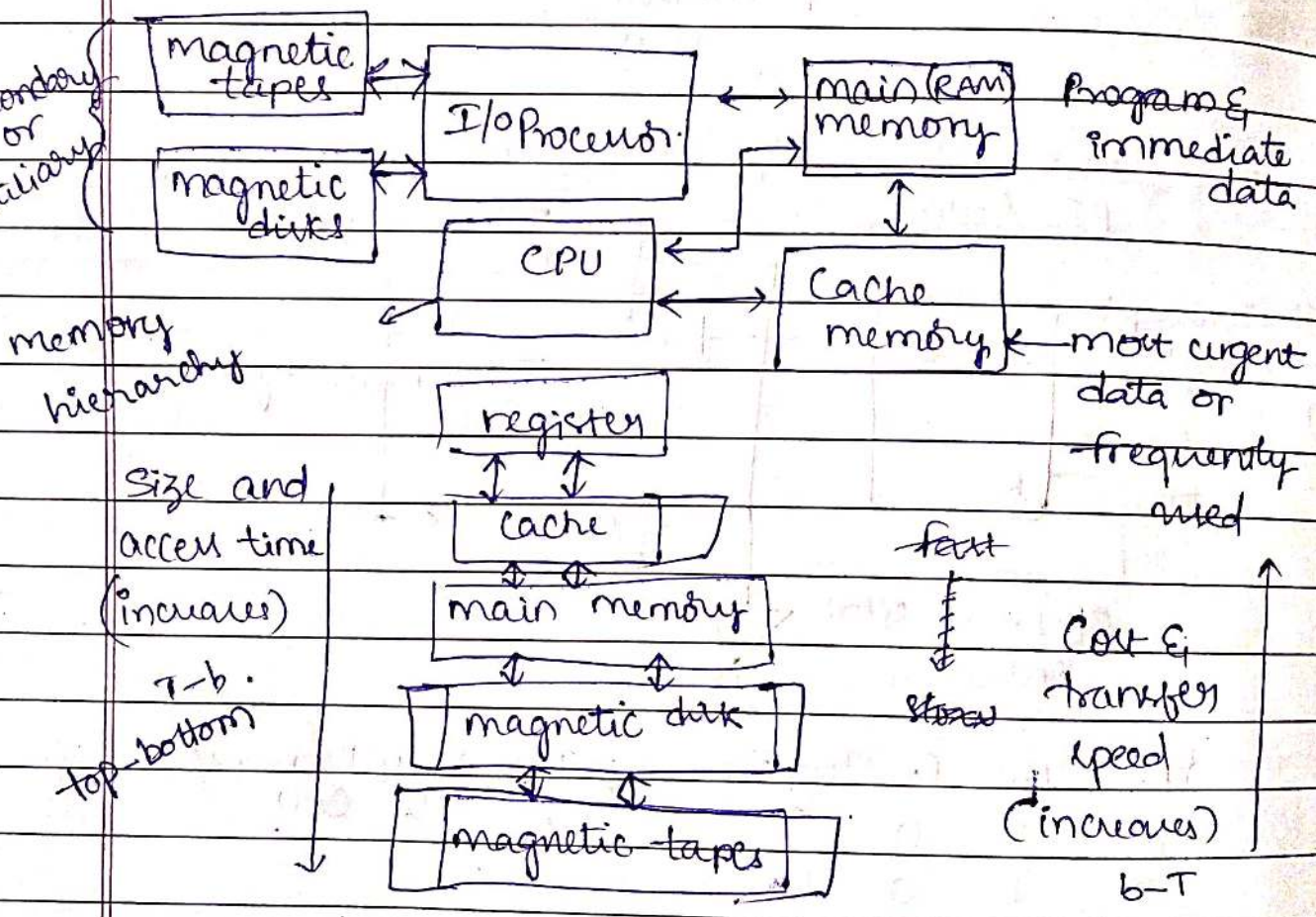
PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

UNIT-05 (MEMORY UNIT)

Date 05/04/23
Page No. 1000000

→ Memory unit essential component since used for storing data & Programs.
→ The memory unit that communicates directly with CPU — Main memory. (RAM) ⇒ volatile (temp)
→ Auxiliary memory — Device that Provides
non-volatile (Permanent) ⇐ { backup storage (magnetic disks & tapes).

Memory hierarchy:



Date / /
Page No.

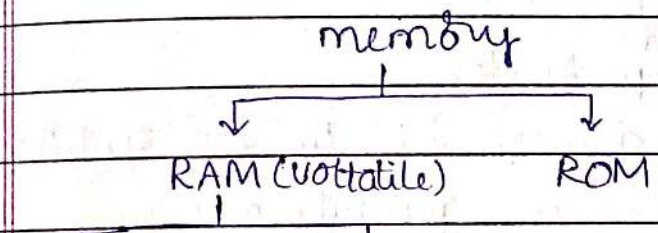
Cache Memory: Special very-high-speed memory to increase the Processing speed by making current Program & data available to CPU at rapid rate – Cache RAM.

→ Access time ratio b/w Cache and M/M is about 1 to 7.

→ Access time : Cache : 100ns,
M/M : 700ns

→ Block size : Cache = 1–16 words
Auxiliary = 256 – 2048 words.

Main memory:



Static RAM	Dynamic RAM
- made of flipflops	- made of capacitors.
- MOSFET	- MOS
- ~~No~~ continuous Power supply required	- continuous supply of power not req.
- high Power Consumption	- low power consumption.
- no leakage of currents	- ~~leakage of currents~~
- used for Implementing Cache memory	- ~~used for implementing main memory~~
- costly	- less expensive.

Date 06/04/23
Page No.

ROM

PROM	EPROM	EEPROM
- done once	→ UV rays → Complete disc is Formatted	→ Erasable → Part of disk can be erasable.

☆ 2m **Bootstrap loader:**

→ A Program whose function is to start the Computer Software operating when Power is turned on.

→ The bootstrap loads the Portion of OS from disk to main memory.

→ we use Bootstrap loader

→ and control is transferred to OS which prepare Computer for general use.

→ The ROM Portion of main memory is needed for storing an initial Program is called bootstrap loader.

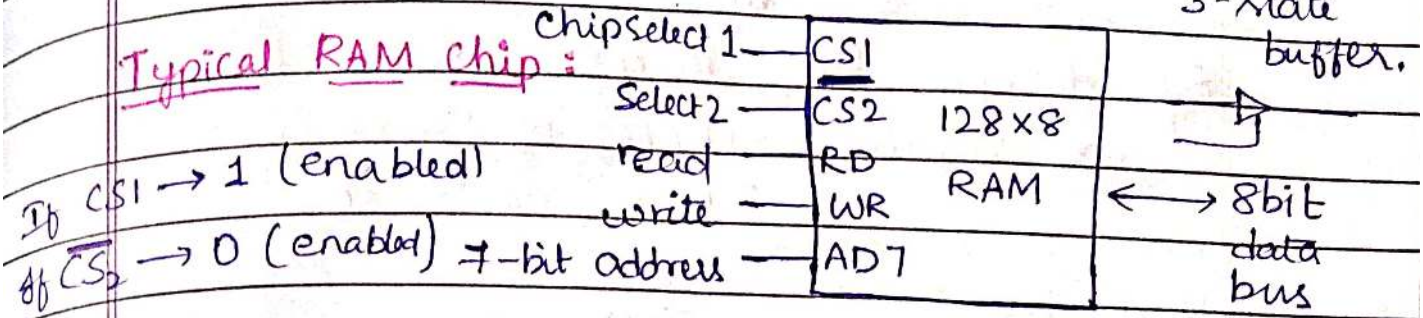
Static RAM → Cache Memory

Dynamic RAM+ROM Chips → Main memory

Date / /
Page No.

Main Memory:

Typical RAM chip:



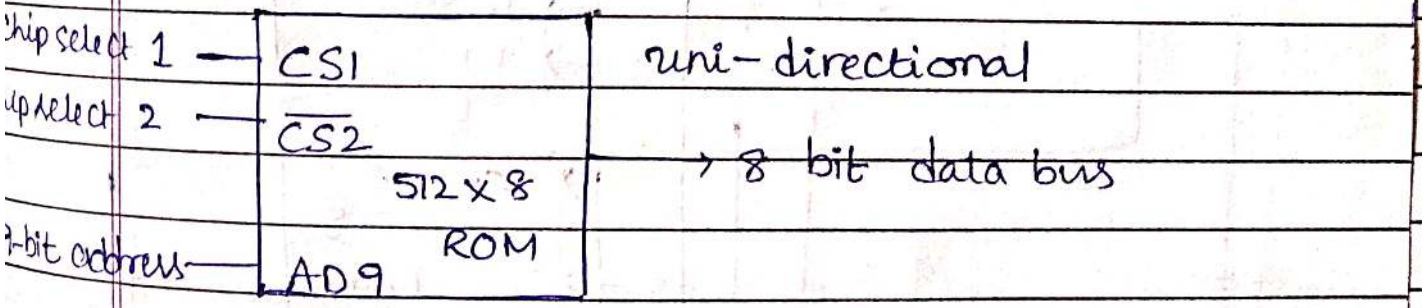
If CS1 → 1 (enabled)
If $\overline{CS_2}$ → 0 (enabled)

CS1	$\overline{CS_2}$	RD	WR	memory fnc	State of data bus
0	0	x	x	inhibit	high-impedence
0	1	x	x	inhibit	high-impedence
1	0	0	0	inhibit	high-impedence
1	0	0	1	write	input data to RAM
1	0	1	x	read	o/p data from RAM
1	1	x	x	inhibit	high-impedence

⇒ 128 × 8 ⇒ there will 7-bit address lines and there will be 8-bit data.
(128 words of data)

* high impedence → (open circuit).

TYPical ROM chip:



⇒ If CS1 = 1 and $\overline{CS_2}$ = 0 then by default it will be a read operation.

⇒ 4 times bigger than the RAM

⇒ For 512×8 ⇒ 9-bit addresses and 8-bits data.

Eg: design for 1024×8 using RAM & ROM

⇒ we require 4 RAM's of 128×8 and 1 ROM of 512×8

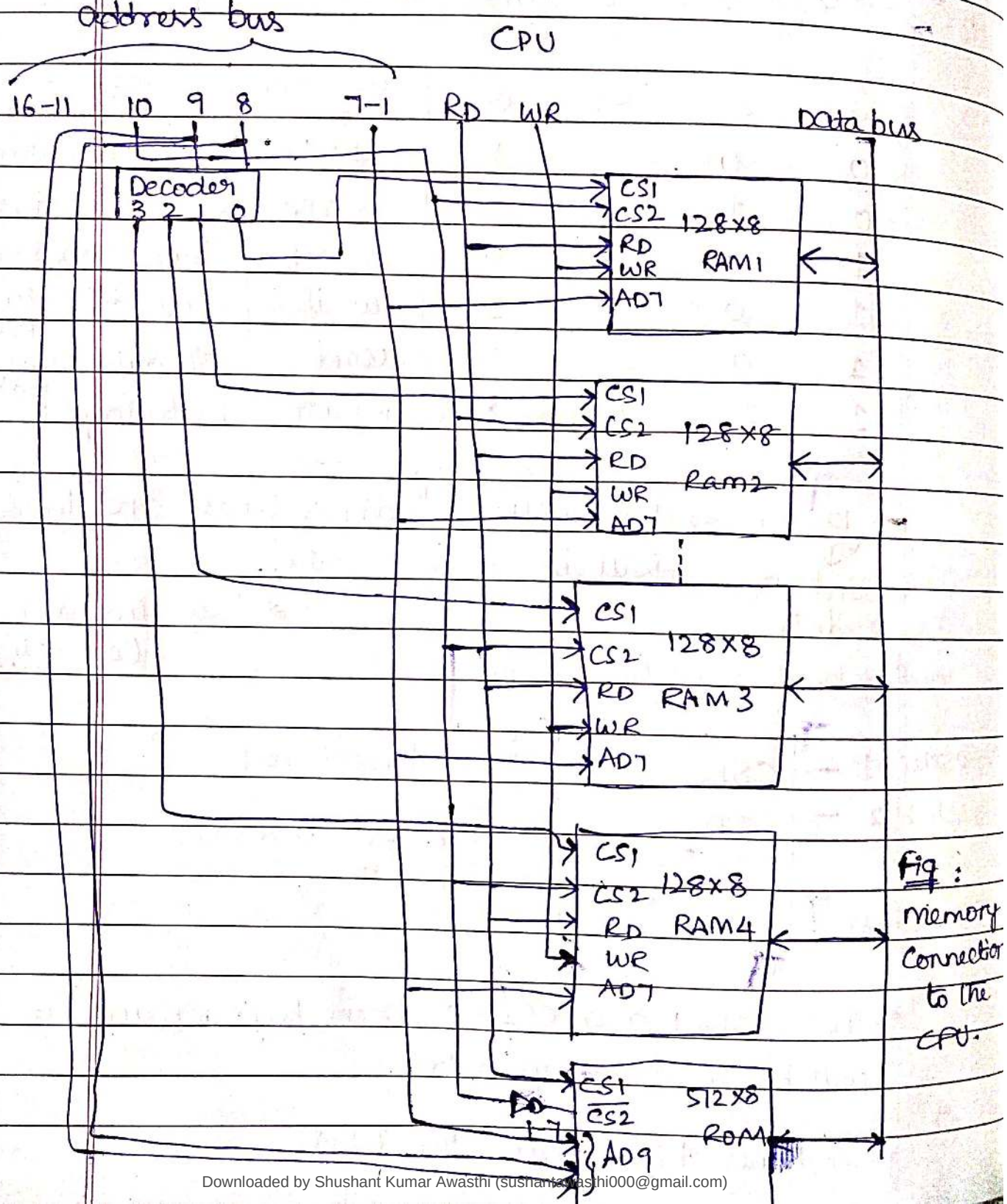
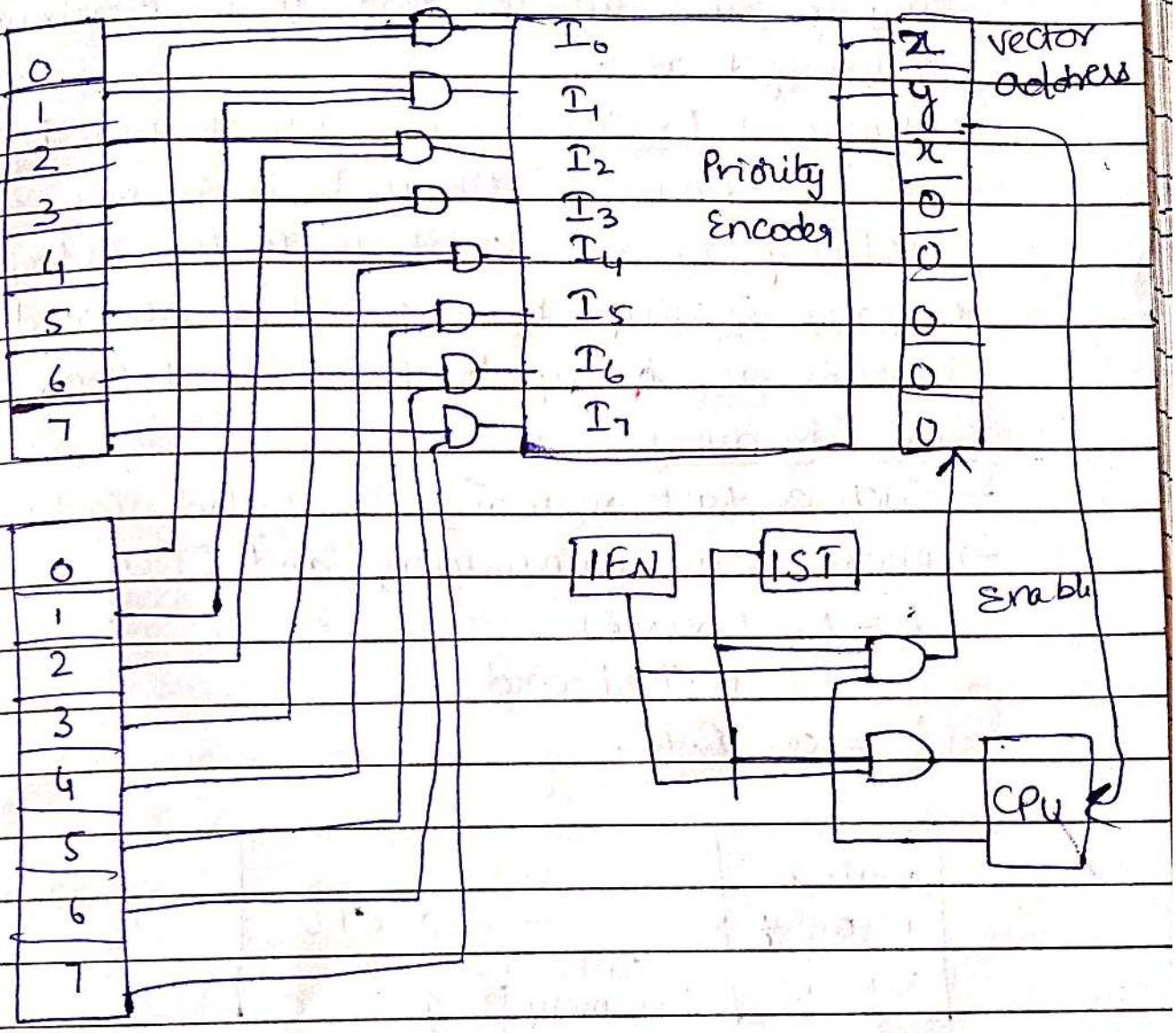


Fig: Memory Connector to the CPU.

Date 07/04/23
Page No.

Q. Design a Parallel Parity interrupt hadware for a system with 8 interrupt sources.



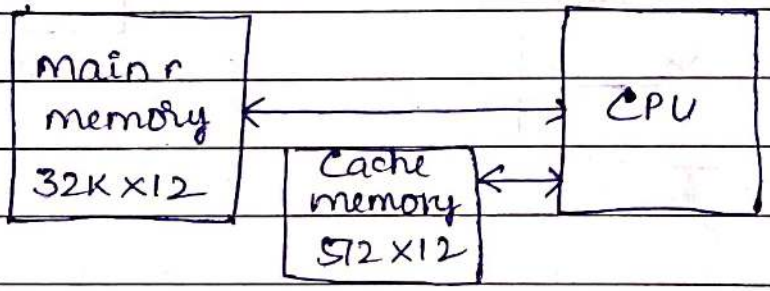
Date 14/04/23
Page No.

Cache Memory:

* Locality of Reference: References to memory at any given time interval tend to be confined within a localized area.
* Temporal Locality: The info which will be used in near future is likely to be in use already.
  → likely to use Eg: Reuse of info in loops.
* Spatial Locality: If a word is accessed near words are likely to be accessed soon.

Cache Memory:

→ Such a fast memory is called cache memory.
→ Placed b/w main memory and CPU
  L_1 = On Processor
  L_2 = on Motherboard
  L_3 = on RAM.



Eg: Example of Cache memory

→ If the word is in cache: Access cache to Provide it to CPU – HIT
→ If the word is not in cache: it is in main memory – MISS

$$\text{HIT ratio (H)} = \frac{\text{No. of hits}}{\text{Hits + miss}} \qquad \text{Miss ratio (M)} = \frac{\text{No. of miss}}{\text{Hits + miss}}$$

$$M = (1-H)$$

Cache Access time

Date / /
Page No.

→ T_c : Access time (cache) → T_M : Main memory access time.

→ Effective time (T_e)

$$T_e = h * T_c + (1-h)(T_M + T_c)$$

Cache Mapping :

→ The transformation of data from main memory to Cache memory - Cache Mapping.

→ Three types of mapping Procedures
* Associative mapping
* Direct Mapping
* Set - associative mapping.

→ Data address have two types : index (9 bits) and tag (6 bits).

Direct Mapping :

→ RAM as Cache - 15 bit address is divided in two fields (tag, index).

disad : Hit ratio is dropped significantly if two or more words whose address have same index but different tags, are accessed repeatedly.

→ The Performance of Cache memory is frequently measured in terms of a quantity called hit ratio.

→ The Cache memory access time is less than the access time of main memory.

Date 18/04/23
Page No.

Virtual Memory:

→ Difference b/w main memory and auxilary memory is Virtual Memory.

→ An address used by Programmer – Virtual address (address space) (Logical add)

→ An address in the main memory – Physical address.

→ The transfer of data from auxiliary to main memory – Swapping.

→ Moving out the unnecessary data – Swap-out.

→ Address in main memory – Memory space or locations. (Physical add).

Address Mapping:

→ Memory mapping table for virtual address → Physical address.

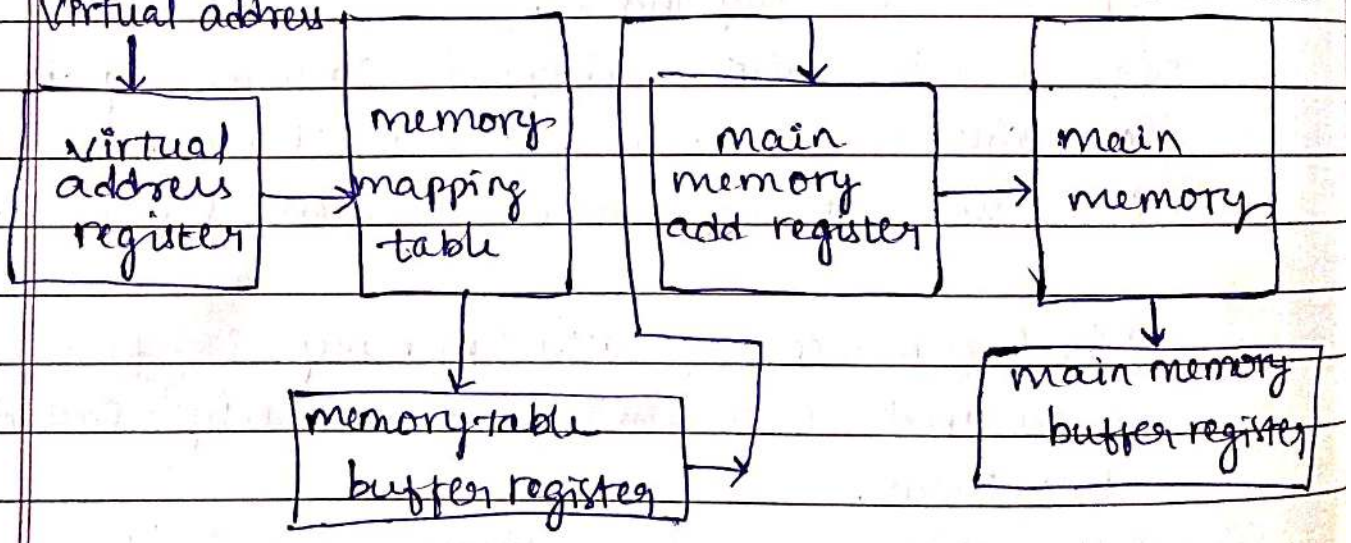


fig: Memory table for mapping a virtual address.

Date 21/04/23
Page No.

Booth's Multiplication Algorithm: (Signed 2's complement)

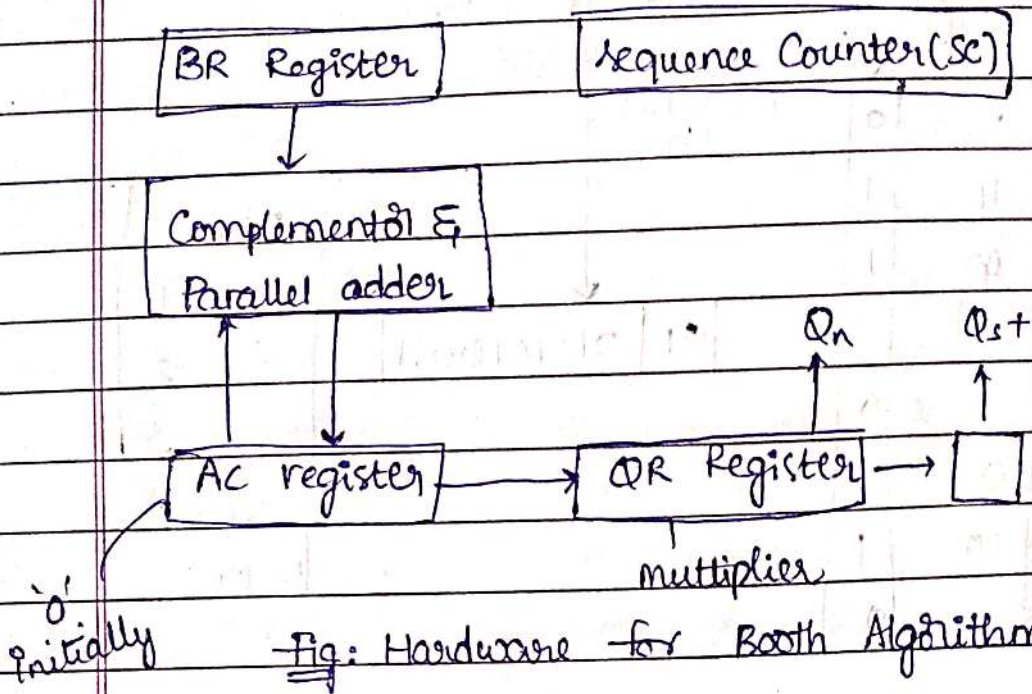


Fig: Hardware for Booth Algorithm.

Q_n	Q_{n+1}	
0	0	ashr AC; QR
1	1	$SC \leftarrow SC+1$
0	1	→ Add $AC \leftarrow AC+BR$
1	0	$AC \leftarrow AC+\overline{BR}+1$

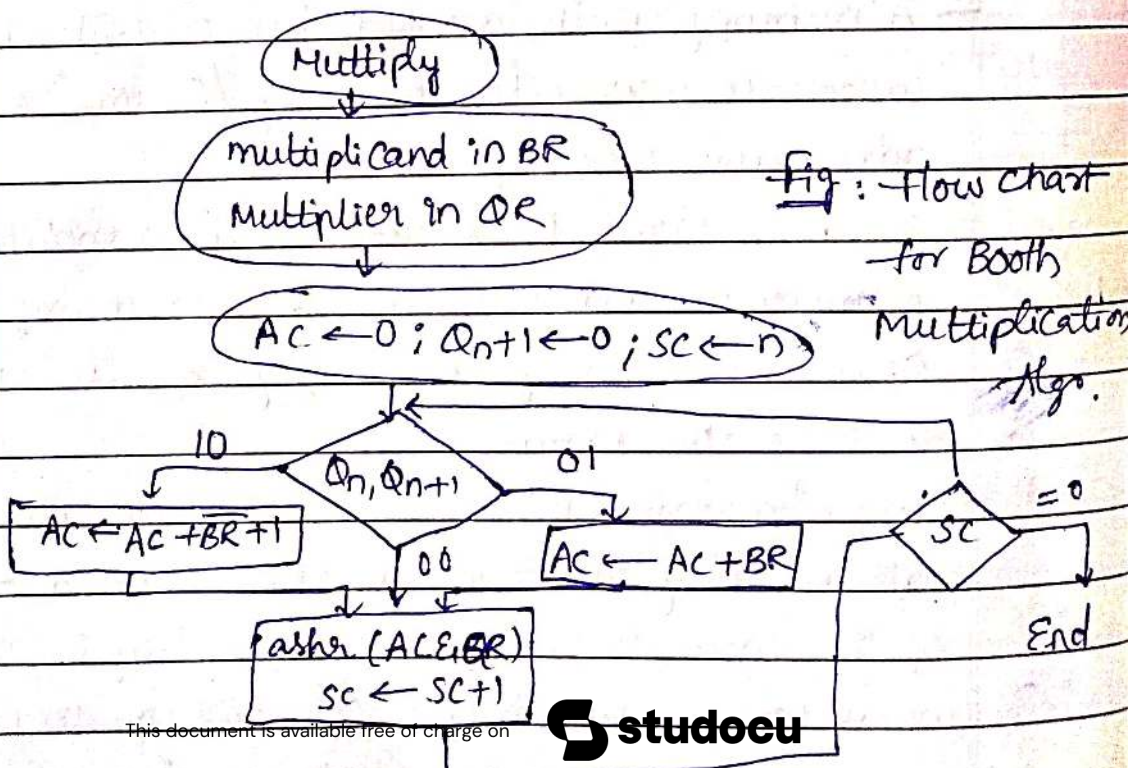


Fig: Flow chart for Booth Multiplication Algo.

Date / /
Page No.

Eg: $\frac{23}{19}$

$\Rightarrow$ 23 → 10111 (BR) ⇒ $\overline{BR}+1$ = 01001

19 → 10011 (QR)

	AC	QR	Q_{n+1}	SC
subtract	00000	10011	0	5 (101)
	01001			
	01001	10011	0	5 (101)
ashr, SC ← SC+1	00100	11001	1	4 (100)
Q_{n+1} = 11	00010	01100	1	3 (011)
ashr, SC ← SC−1	10111			
Q_{n+1} = 01,	11001	01100	1	3 (011)
add BR.				
ashr, SC ← SC−1	11100	10110	0	2 (010)
Q_{n+1}, Q = 00	11110	01011	0	1 (001)
ashr, C ← SC−1	01001			
Q_{n+1} = 10,				
sub AC ← AC+BR+1	00111	01011	0	1 (001)
dropped				
shr, SC ← SC−1	00011	10101	1	0 (000)

$\left.\begin{matrix}00\\11\end{matrix}\right\}$ → ashr, SC ← SC−1

01 → add (AC ← AC+BR

10 → Add (AC ← AC+$\overline{BR}$+1

Date 25/04/23
Page No.

BR = 10111 → 2's Complement
10110 → 1's Complement
11001 → -9 → multiplicand

Eg: BR = 010111
$\overline{BR}+1$ = 101001

	Q_n	Q_{n+1}	AC	Q_R	Q_{n+1}	SC
			000000	010011	0	6
	1	0	101001			
ashr, SC←SC-1			101001	010011	0	6
			110100	101001	1	5
	1	1	111010	010100	1	4
			010111			
			010001	010100	1	4
ashr, SC←			001000	101010	0	3
ashr, SC			000100	010101	0	2
	1	0	101001			
			101101	010101	0	2
ashr, SC←			110110	101010	1	1
add BR			010111			
			001101	101010	1	1
ashr SC←			000110	110101	0	0