Central Processing Unit

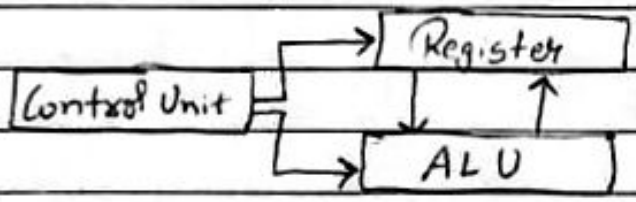
→ Part of Computer that performs all data-processing is called CPU.

↳ CPU = Registers + ALU + Control Unit
(Arithemetic & logical Unit)

Register :- It stores the intermediate data that is used during the execution.

ALU :- It generally performs necessary microoperations to execute the instruction.

Control Unit :- Supervise transfer of inf. into register & instruct ALU what operation to perform.

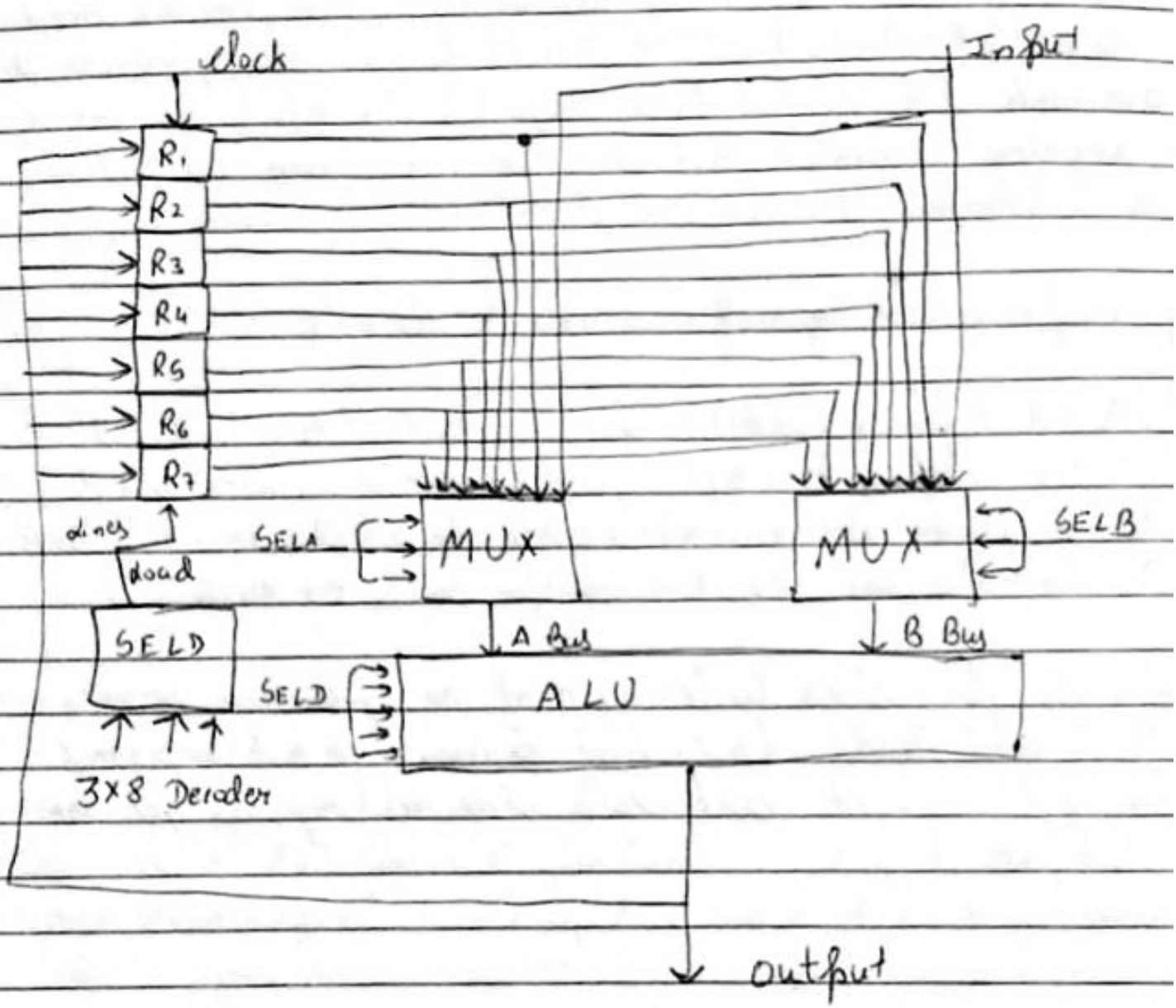


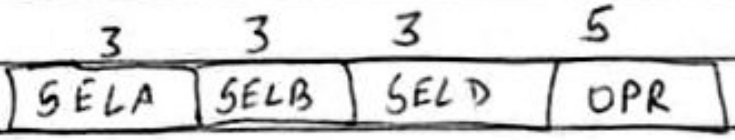
General Register Organization :-

Memory locations are needed for various things like storing points, return address, store temp data. Hence refer to memory for such operations is very time consuming because memory accss is the most time consuming operation.

So to avoid it is more convenient to store these intermediate values in processor register where a large no of registers are there in CPU it is most efficient to connect them with a common bus.

Register communicate with each via common bus to perform operations like data transfer, performing various microoperations.





A bus organization for seven CPU register. Each register is connected with two multiplexers (MUX) to form the two buses A & B.

The selection lines of each MUX select one register or input data for the particular bus.

The A and B Bus is connected to a ALU. The operation select in the ALU tells us which microoperation need to be performed. The result of that microoperation is available for output data as well as input for other registers. Register that recieve inf from output bus is selected by a 3×8 decoder.

eg Perform the operation $R_1 \leftarrow R_2 + R_3$

① MUX A Selector (SELA) :- Place content of R_2 into bus A
② MUX B Selector (SELB) :- Place content of R_3 into bus B
③ ALU operation Selector (OPR) :- Provide Arithmetic addition A+B
④ Decoder Selection (SELD) :- Transfer output o to R_1

In a single clock cycle each of the operation is performed and output send as input to all registers so that in next clock cycle selected registers get the output data.

Stack Organization

A useful feature included in most of the CPU is stack or last-in first out list. It is a storage device that stores data in form that item stored last is first retrieved

The Stack in digital Computer is a memory Unit with an address register. The register holds the address of the stack is SP (Stack pointer) because its value always point to the top of stack.

Two operation of stack are

a) PUSH:- The operation of insertion is push. Insert at top

b) POP:- The operation of deletion is pop Remove from top

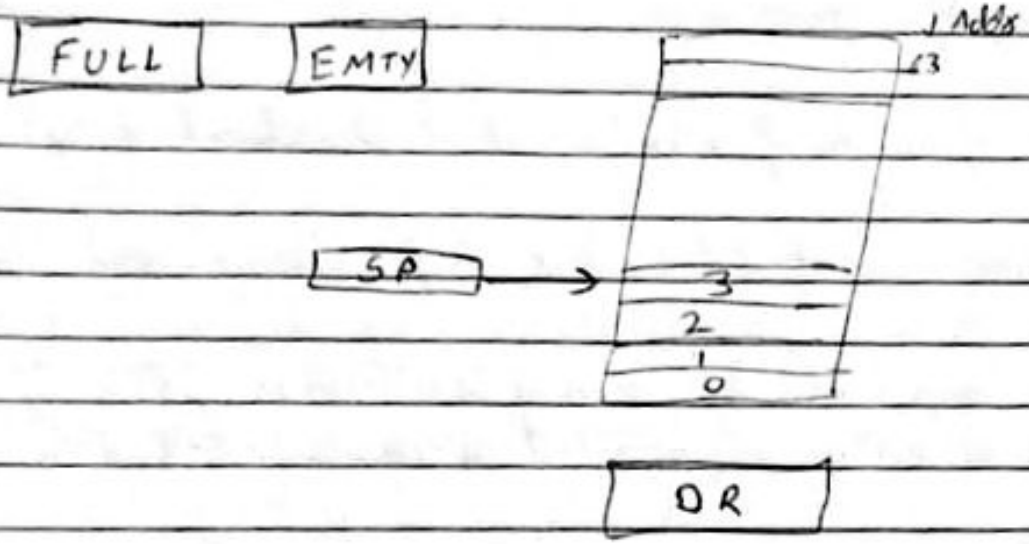
However in ~~Stack~~ Computer Stack nothing is push or pop only Stack Counter is either incremented or decremented.

Register Stack

↳ Collection of finite no. of memory words or register Here the Organization of a 64 word register Stack. The Stack pointer SP Contain a binary number whose value equal to the add of word that is Currently top of Stack.

Three items are pushed into Stack A, B, C. So Item C is the top of Stack So SP is now 3. To remove top item we simply decrement SP. Now Item B is the top item now lets say we need to add a element we simply a Increment SP but here we see we dont actually remove C but it doesnt matter as that C is ever written by that new value

In a 64-word Stack SP contain 6 bits because $2^6 = 64$. So it cannot exceed with 63 (111111) So it we either increment 1 in (111111) or decrement 1 from (000000) to handle that We have two flags



If the Stack is FULL a binary Set to 1
If the Stack is empty EMTY Set to 1

DR Contain binary data to be written or read out of the Stack.

Push operation: Initially SP is cleared to 0, EMTY set as 1 and FULL set as 0. It means Stack is marked empty.

So if Stack is not full (FULL==0) then new item is inserted into Stack

(1) SP ← SP+1 (Increment Stack pointer)
(2) M[SP] ← DR Write item on top of the Stack pop
(3) If (SP==0) then (FULL ← 1) check if stable is full
EMTY ← 0

Pop operation: A element is deleted from the Stack if Stack is non-empty ~~(If EM~~

(1) DR ← M[SP] Read item from top of Stack
(2) SP ← SP-1 Decrement SP
(3) If (SP==0) then EMTY ← 1 Check if Stack is empty
(4) FULL ← 0 Mark Stack not full

Instruction Format

A Computer has Variety of instruction Codes formats. It is func. of Control Unit within the CPU to interrupt each instruction code & necessary control functions needed to process the instructions. Format of an instruction is usually a rectangular box Symbolizing the bits of the instruction

Most common fields found in instruction format are

1. An operational code fields that specifies the op to be performed
2. An address field that designates a memory address or processor register.
3. A mode field that specifies the way the operand or the eff address is determined.

Register address

Operans residing in memory are specified by their memory address. Operans residue in processor register are specified with a reg. address. Register add is a binary number of k bits that defines one of 2^k reg. in CPU.

CPU with 16 process register $R_0 - R_{15}$ will have reg. address for 4 bits.

No. of add fields depends on internal Organization of registers.

1. Single accumulator Organization
2. General register Organization
3. Stack Organization

Single accumulator Organization

All instructions performed with an implied accumulator register. Uses one address field.

ADD instructions in this case result in the operation

$AC \leftarrow AC + M[x]$

↓ ↳ a Memory Word located at add. x

Accumulator Reg.

General Register Organization

Needs 3 register add fields

ADD R_1, R_2, R_3

$R_1 \leftarrow R_2 + R_3$

No. of add. fields can be reduced from 3 → 2, if destination reg. is same as one of input register

ADD R_1, R_2

$R_1 \leftarrow R_1 + R_2$

Computer with most multiple processor register use the move instruction with a MOV to symbolize a transfer instruction.

Stack Organization

→ Here we has push & pop operation require an add field.

Instruction PUSH x push word at address x to the top of stack
Stack pointer updated automatically.

Operation-type instruction don't need add field in stack.

Inst ADD in stack Uses no add field.

There is no need to specify operands with an address field since all operands are implied to be in the stack

Three-Address Instruction

Here add field use to specify process reg. & memory operand.

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

ADD	R_1, A, B	$R_1 \leftarrow M[A] + M[B]$
ADD	R_2, C, D	$R_2 \leftarrow M[C] + M[D]$
MUL	X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

We assume Computer has two process reg. R_1 & R_2

Two address Instructions

Most common in commercial computers

$X = (A+B) * (C-D)$

MOV	R1, A	R1 ← M[A]
ADD	R1, B	R1 ← R1 + M[B]
MOV	R2, C	R2 ← M[C]
ADD	R2, D	R2 ← R2 + M[D]
MUL	R1, R2	R1 ← R1 * R2
MOV	X, R1	M[X] ← R1

MOV instruction is used to transfer operand to & from memory & process registers.

One address Instruction

Uses implied accumulator (AC) for all data Manipulation

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

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

All operations are performed b/w acc. & memory operand. T is a temporary memory location

Zero - address Instruction

Stack Organization Computer does not take add field for ADD & MUL.

The PUSH & POP inst need an add field to specify the operand the communicates with stack.

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

TOS (Top of stack)

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

Addressing Modes

Operation field in an instructions specifies operation to be performed. Operation must be executed on some data or memory add. Way operands are choose during program execution depend on addressing mo

Addressing mode specifies a rule for interpreting or modifying the add field of the instructions before the operand is actually referred.

Refer add Mode

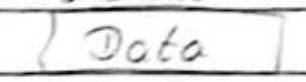
An instruction to cycle has 3 main steps

1. fetch instruction from memory
2. Decode instruction
3. Execute instruction

Types of addressing Mode

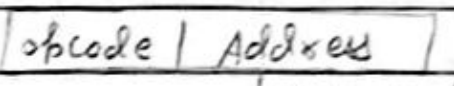
1. Implied Mode:-

Operand is specified implicitly in the defination of instruction
Used for Zero & One add field



2. Immediate Mode:-

Operand is specified in the instruction. In this instruction has an operand field rather than add field. The operand field contains actual operand to be used in conjuction.

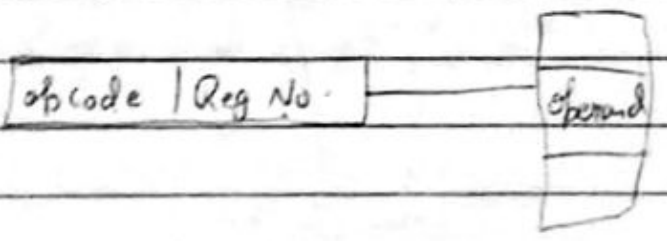


This data is gen.
↳ Data is directly stored here (constant)

3. Register Mode:-

Operand is present in the register

Register no is stored / written in instruction

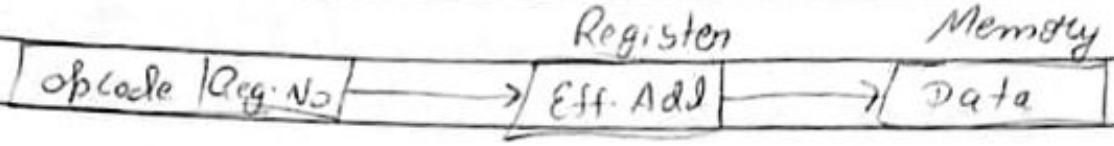


Date: __/__/__

MON TUE WED THR FRI SAT SUN

Register Indirect Mode

Here register contain add of operand rather than operand itself

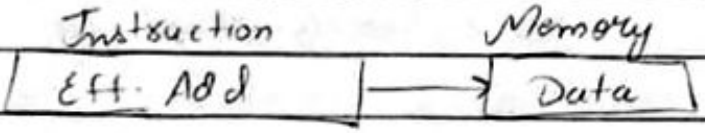


Auto increment & Autodecrement Mode

It is similar to register indirect mode except that register is ↑ or ↓ after or before its value is used to access memory. It is necessary to ↑ or ↓ the register after every access. This achieved using ↑ or ↓ instruction

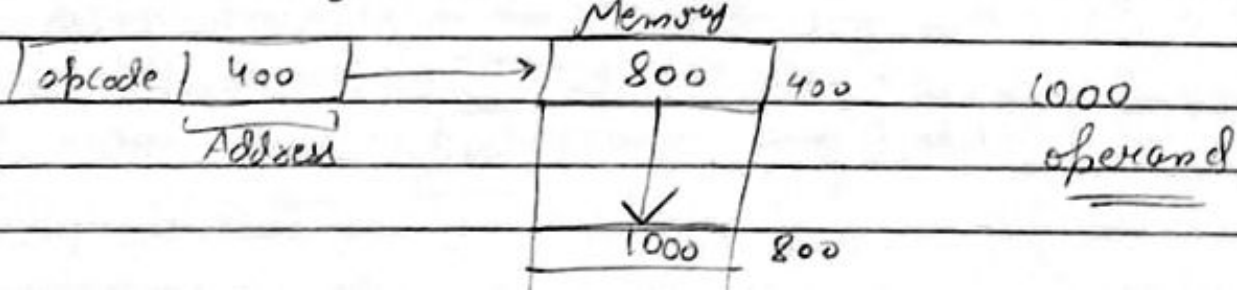
Direct Address Mode:-

Actual address is given in instructions
Use to access Variable



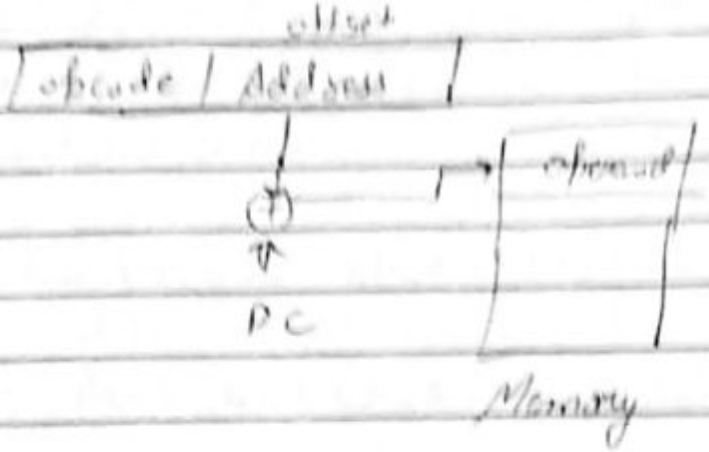
Indirect Address Mode:-

→ Used to implement pointers & passing par.
→ 2 Memory access req.



Relative addressing Mode

Effective addr = PC + offset



Computer Instructions

→ Data transfer Instruction
→ Data Manipulation Instruction
→ Program Control Instruction

a) Data transfer Instructions

↳ Move data from one location to another without changing the data content.

Load (LD) :- Transfer from memory to process reg.
Store (ST) :- Transfer from process reg to memory
Move (MOV) :- Transfer from one to another reg.
Exchange (XCH) :- Swaps info b/w two reg.
Input (IN) }
Output (OUT) } Transfer among reg input or output.

Push } Transfer b/w processor reg & a memory stack.
Pop }

Data Manipulation Instruction

Data manipulation instruction perform operations on data & provide the computational capabilities

Divided into 3 parts

a) Arithmetic instruction ADD, SUB, MUL, DIV

b) Logical & bit Manipulation CLR, AND, OR, XOR

c) Shift instruction SHR, SHL, ROR, ROL

Program Control Instruction

Instructions are always stored in successive memory locations.

When processed in CPU the instruction is fetch from consecutive memory location & executed.

Each time a instruction is fetched program counter is incremented so it contains add of next instruction.

After data trasfer & execution control returns to fetch cycle.

On other hand program control instruction is a type of instruction when executed may altered by flow of execution.

Program Counter instruction modify & change flow of program. It alters seq of program execution.

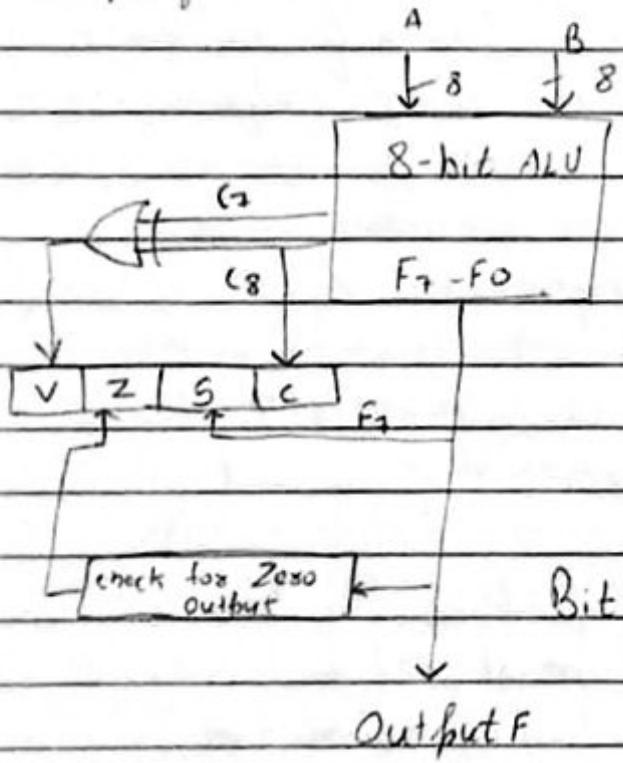
Branch (BR) & Jump (JMP) inst may be conditional or Un conditional, when executed the branch inst cause a transfer of value of ADR into PC

Condition Skip (SKP) skip next instruction if code match.

Status Bit Conditions (Not that)

It is also called Condition-Code bits or flag bits

The four status bit are symbolized by C, S, Z, V
The bits are set or cleared as a result of an operation performed in the ALU.



Bit C (carry) is set to 1 if the end carry C8 is 1. It is cleared to 0 if the carry is 0

Bit S (Sign) is set to 1 if highest order bit F is 1. It is Zero if bit is Zero.

Bit Z (Zero) is set to 1 if the output of ALU contain all Zeros. It is cleared to 0 otherwise

Refer Data Tsns Slide Last

Date: __/__/__
MON TUE WED THU FRI SAT SUN

Bit (v) overflow: It is set to 1 if the exclusive-OR of the ~~bo~~ last to carries is equal to 1 & cleared to 0 otherwise

Control Branch Instructions

A Branch is an instruction in a Computer program that can cause a Computer to begin executing a diff instruction & thus deviate from default behaviour.

Program Interrupt

↳ Use to handle Variety of problems arise out of normal program sequence.

Transfer of program control from a currently running program to another program as a result of external or internal request.

Control return to original program.

Interrupt is usually initiated by an internal or external signal rather than from the execution of an inst.

After the program has been interrupted & the service routine been executed, the CPU must return to exactly the same state that it was when the interrupt occured.

Date: __/__/____

MON TUE WED THR FRI SAT SUN

The State of the CPU at the end of the executor cycle is determined from:-

1. The Content of all program Counter
2. The Content of all process register
3. The Content of Certain Status Conditions

Program Word Status:- The Collection of all Status bit Conditions in the CPU is Sometimes called a program Status word or PSW.

Types of interrupts:-

1. External interrupts:- Came from I/o devices, from a timing device etc

eg I/o device requesting, transfer of data, power failure.

2. Internal interrupts:- Internal interrupts arise from illegal or erroneous use of an instruction or data. It is also called traps.

eg attempt to divide by Zero, register overflow, Stack Overflow

3. Software interrupts:- External & Internal interrupt are initiated from Signal occur in hardware of the CPU.

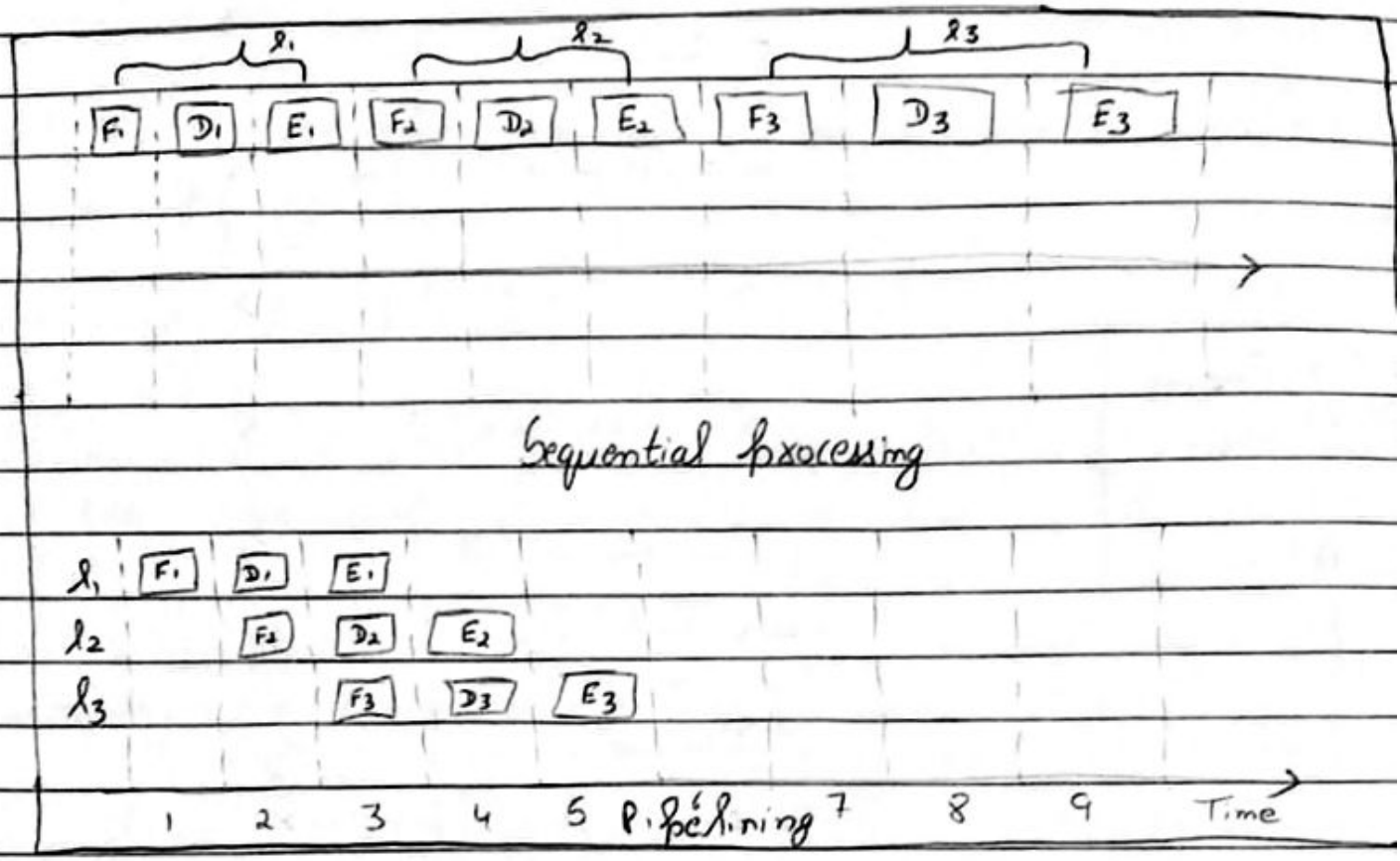
A Software interrupt is initiated by executing an inst. Software interrupt is a special call instruction that behaves like an interrupt rather than a Subroutine Call.

CISC	RISC
1) Complex Instruction Set Computer	Reduced instruction Set Computer
2) Large no. of instructions	Less no. of instructions
3) Variable length inst format	Fixed length Inst format
4) Large no. of addressing modes	Few no. of AM
5) Cost is high	Low Cost
6) More powerful	Less powerful
7) Several cycle inst	Single cycle inst.
8) Manipulate directly into memory	Only in register

Pipeline & Parallel processing

Pipelining is a technique of breaking a sequential process into sub-segments or sub-operations. Each sub-process takes place in a dedicated segment together with all other segments

Pipelining Vs Sequential processing



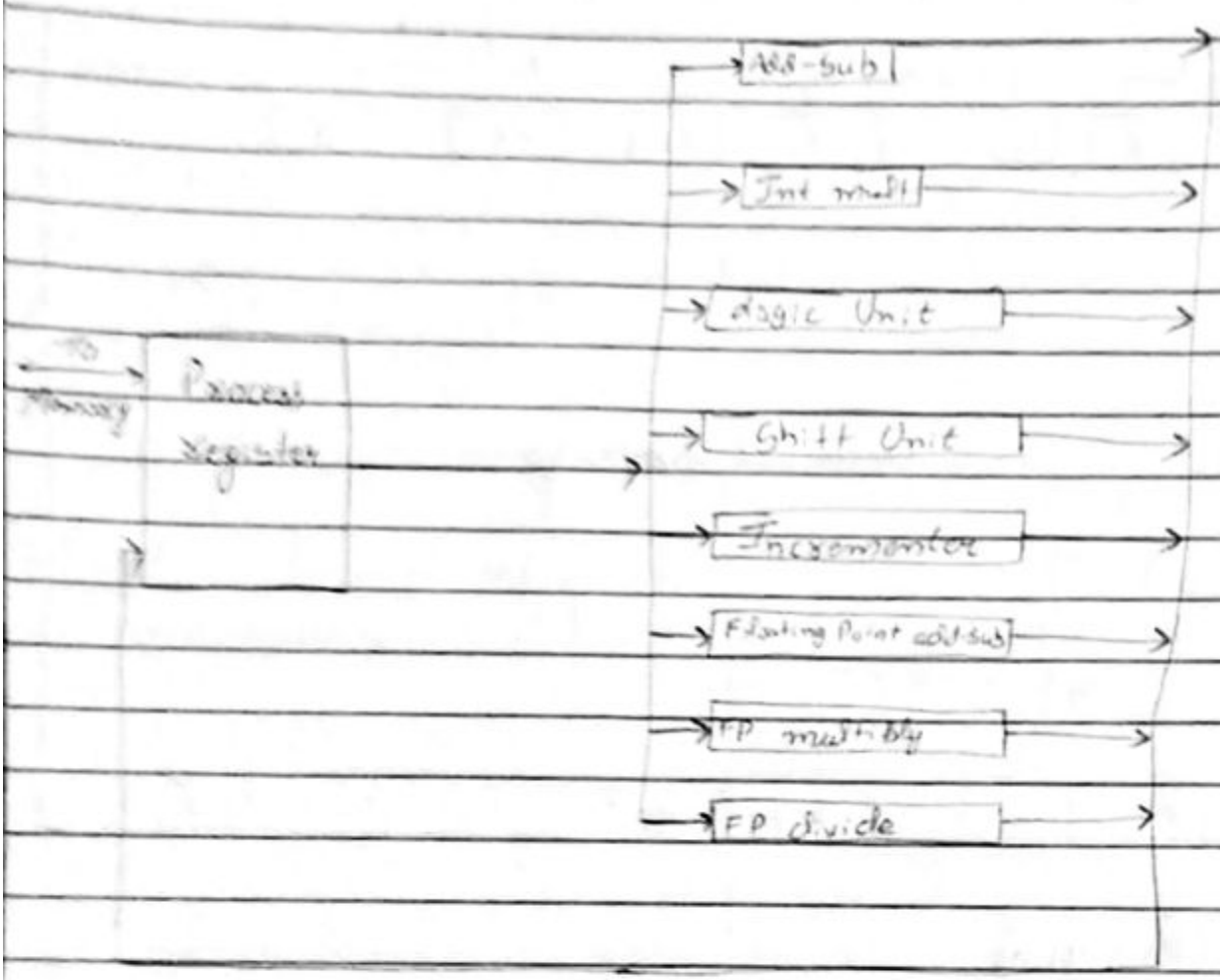
Parallel Processing

Term used to denote a large class of techniques that are used to provide simultaneous data-processing increase computational speed

Instead of processing each instruction sequentially as in a conventional computer a parallel processing system is able to perform concurrent data processing.

System has two or more ALU able to execute two or more instructions.

Multiple functional Points



Way of seperating the execution unit into eight functional Units operating in parallel.

Date: __/__/____

MON TUE WED THUR FRI SAT SUN

The operands in the register are applied to one of the units depending on the operation specified by the instruction associated with the operand.

The operation performed in each func. unit is indicated in each block of the diagram.

Adder & Integer multiplier perform arithmetic operation with integers.

Floating point operations are seperated into 3 circuits for parallel operation.

Flynn's Classification

Organization of a Computer System by the no. of instructions & data items that are manipulated simultaneously.

Normal operation of a Computer is to fetch inst. from memory & execute them in processor.

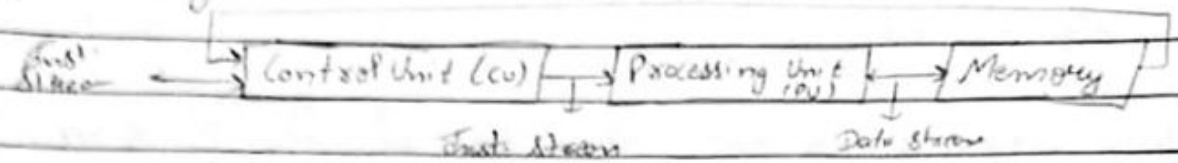
Seq. of inst fetch from memory form a inst. Stream.

Operations performed on the data in the processor constitutes a data Stream.

Flynn Divide Computer into 4 major groups

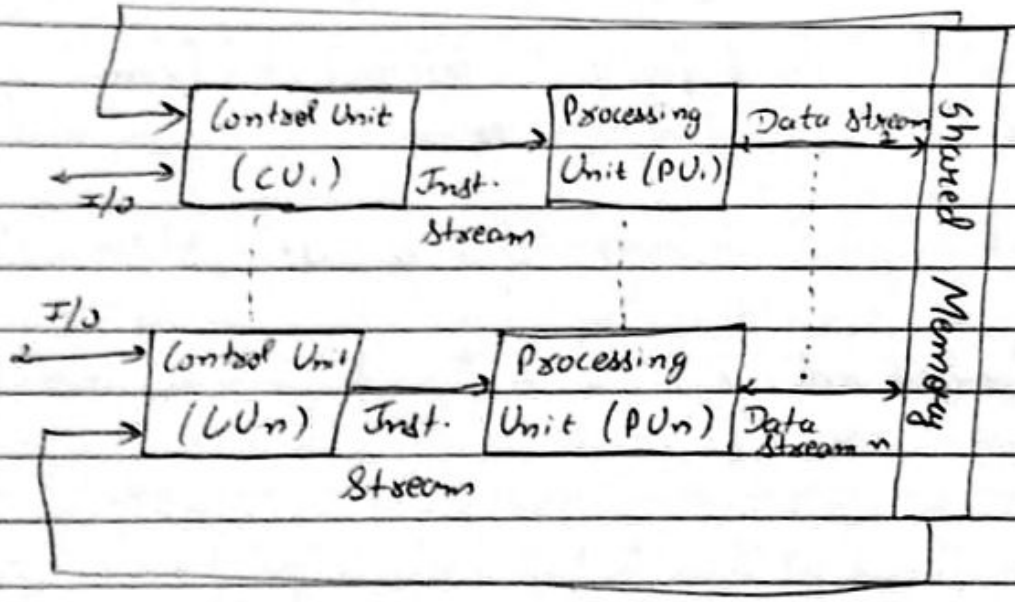
① Single Instruction Stream, Single data Stream (SISD)

Single Computer, Contain Single Control Unit, processor Memory Unit. Instruction are executed Sequentially and Systems may or may not have internal parallel processing



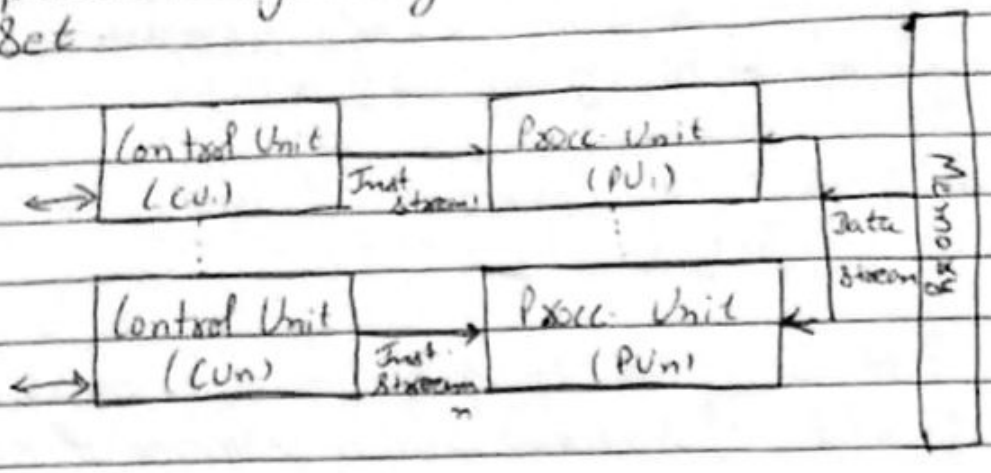
② Single Instruction Stream, Multiple data Stream (SIMD)

Many processing Unit under the supervision of a common control unit. All processor recieve the same inst. from the control Unit but operate on diff. items of data.



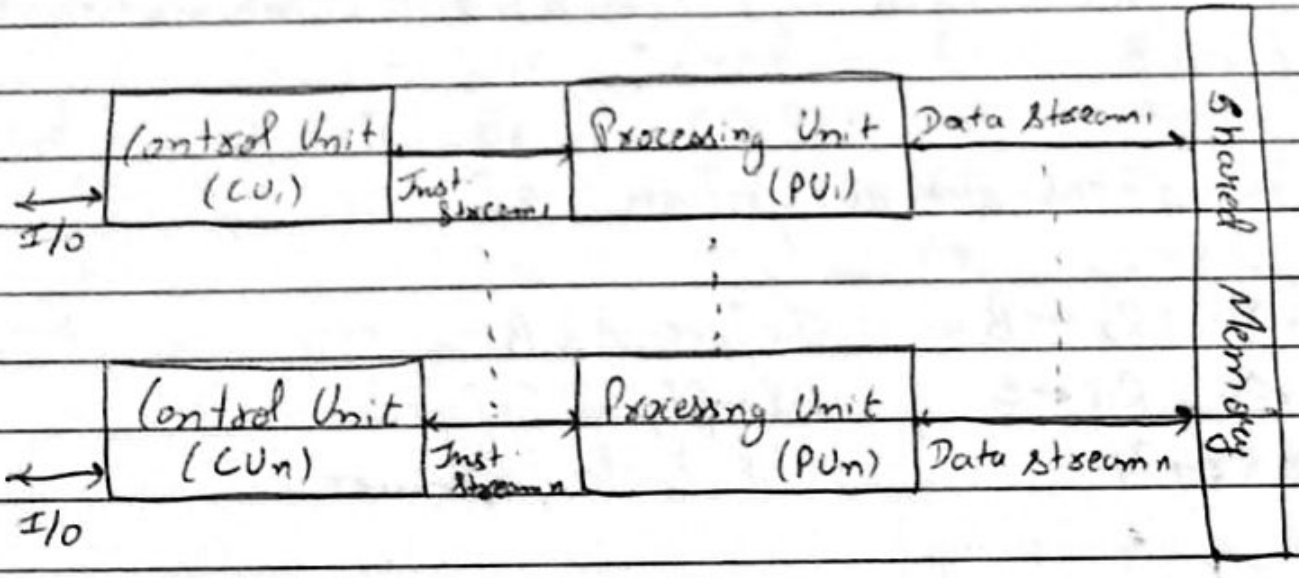
③ Multiple Instruction Stream, Single data Stream (MISD)

No of processing Units performing diff ~~inst~~ operations by using diff inst. on the same data set



④ Multiple Instruction Stream, Multiple data stream

System capable of ~~performing~~ processing several program at the same time Most multiprocessor & multi Computer System.



Pipelining is a process of decomposing a seq. process into sub operations with each sub process being executed in a special dedicated segment.

A pipeline is visualised as collection of processing segments through which binary inf flows.

Result obtained from each segment transfer to next segment in the pipeline.

eg: Suppose that we want to perform the combined multiply & add operation with a stream of no.

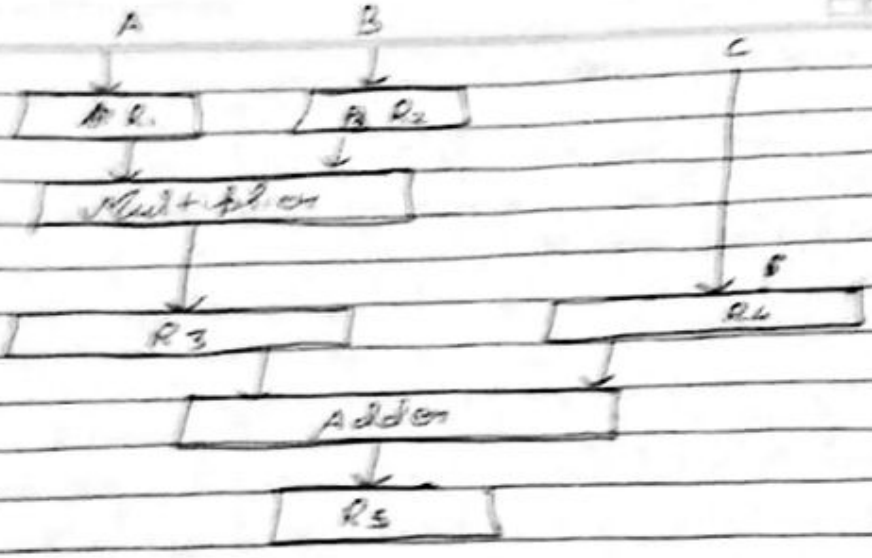
$$A_i \times B_i + C_i \quad i = 1 \rightarrow 7$$

Each sub-operation can be implemented in a segment with a pipeline

Each segment has two reg. and a combinational circuit.

Sub operations are as follows

$R_1 \leftarrow A$, $R_2 \leftarrow B$	Input A & B
$R_3 \leftarrow R_1 \times R_2$ $R_4 \leftarrow C$	Multiply & Input C
$R_5 \leftarrow R_3 + R_4$	Add C to product



Clock pulse No	Segment 1 R_1	Segment 1 R_2	Segment 2 R_3	Segment 2 R_4	Segment 3 R_5
1	A_1	B_1			
2	A_2	B_2	$A_1 \times B_1$	C_1	
3	A_3	B_3	$A_2 \times B_2$	C_2	$A_1 \times B_1 + C_1$
4	A_4	B_4	$A_3 \times B_3$	C_3	$A_2 \times B_2 + C_2$
5	A_5	B_5	$A_4 \times B_4$	C_4	$A_3 \times B_3 + C_3$
6	A_6	B_6	$A_5 \times B_5$	C_5	$A_4 \times B_4 + C_4$
7	A_7	B_7	$A_6 \times B_6$	C_6	$A_5 \times B_5 + C_5$
8			$A_7 \times B_7$	C_7	$A_6 \times B_6 + C_6$
9					$A_7 \times B_7 + C_7$

General Consideration

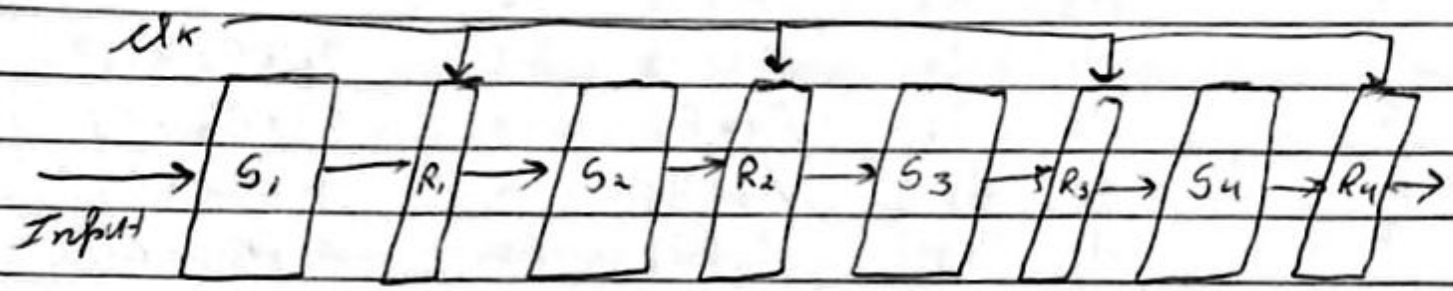
Any operation that can be decomposed into sub operation of some complexity can be done under pipeline processor.

General structure of four segment pipeline

→ The operands pass through all four segments in a fixed seq. Each segment contain a combinional circuit S_i ~~are~~ a sub-operation over the data stream flowing through the pipe.

→ The segment are seperated by reg. R_i that holds the intermediate result

Task is defined as the total operation performed going through all segments in pipeline.



The behaviour of such pipeline is illustrated with a space-time diagram

Space time diagram for four Segment task from $T_1 \rightarrow T_6$

Segment ↓ / Clock →	1	2	3	4	5	6	7	8	9
1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

To Complete n task with K-Segment pipeline req $K+(n-1)$ clock cycles

Axithmetic Pipeline

Pipeline axithmetic Units are usually found in very high speed Computers. They are used to implement F-P operations, multiplications of F-P numbers

Peripheral Devices

Devices that are under direct control of the computer are said to be connected on-line. The devices are designed to read info into or out of the memory unit upon command from the CPU.

I/o devices connected to computer are called peripheral devices.

Monitor & Keyboard

Video monitor are the most commonly used peripherals. They consist of a keyboard as the input device & a display unit as the out put device.

Date: __/__/____

MON TUE WED THU FRI SAT SUN

Printer

Printer provide a permanent record on paper of computer output data or text

Three
↳ Daisy wheel Printer
↳ Dot Matrix "
↳ Laser printer

Magnetic tape & Magnetic disk

Magnetic ~~disk~~ tapes are used most for storing files of data for eg a Company payroll record

Mag disks have high-speed rotational surfaces coated with magnetic material.

I/O Interface

I/o interface provide a method for transferring inf. b/w internal storage & external I/o.

Peripherals connected to a computer need special communication links for interfacing them with the CPU.

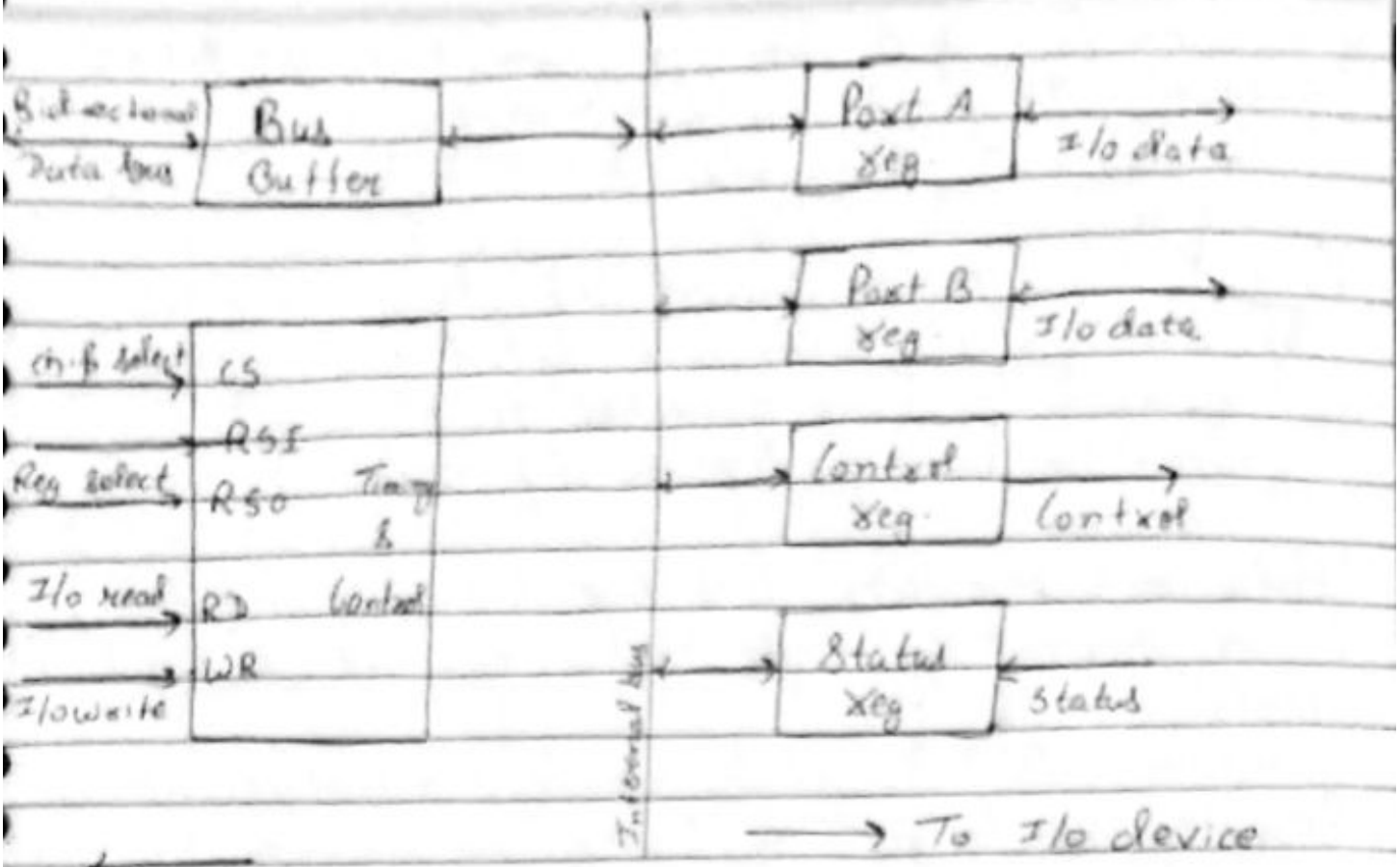
Major differences are :-

a) Peripherals are electro mechanical & electromag. devices & their manner of op. is different from the op. of the CPU & memory.

b) Data transfer rate of peripherals is usually slower than the CPU.

c) Data codes & formats in peripherals differ from word format in CPU.

d) Op. modes of peripherals are diff from each other & must be controlled.



In this we have two data reg called ports. a control reg, a status reg., bus buffer & timing & control.
It communicate with CPU through data bus.

The chip select & register select determine address assigned to interface. The I/o read & write are two lines specially for input & output

The I/o data to & from the device can be transferred into either port A or port B.

If interface is a printer then it only serves output data b.c it is a character reader then only serves input data.

Control reg recieves control inf from the CPU. By loading appr bits into control reg. The interface & I/o device attached to it can be placed in variety of operating modes

Bits in status reg used for status codes & for recording errors that may occur during data transfer.

Asynchronous Data Transfer

Internal operations in a digital system are synchronized by a common clock pulse supplied by a common pulse generator.

Clock pulse are applied to all reg within a unit & all data transfer within reg occur simultaneously during the occurence of a clk pulse.

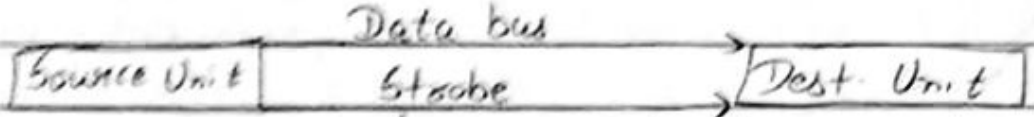
Two units such as CPU & I/o interface are designed independently of each other. If the reg in the interface share a common clock pulse with the CPU reg. the transfer b/w two units is synchronous.

The internal timing in each Unit is independent from the other in that each Uses its own private clock for internal reg. The two units are said to be asynchronous to each other.

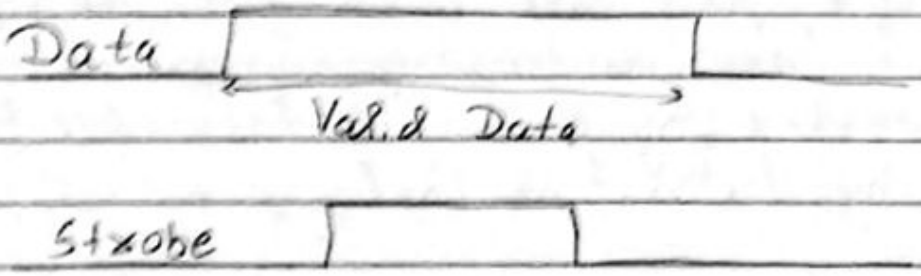
Techniques:-

Asynchronous data transfer b/w two independent Units require control signals be ~~transmitted~~ transmitted b/w connecting links to indicate time at which data is being transmitted.

1. Strobe control:- Strobe pulse supplied to one of the units to indicate to the other Unit when data is send.



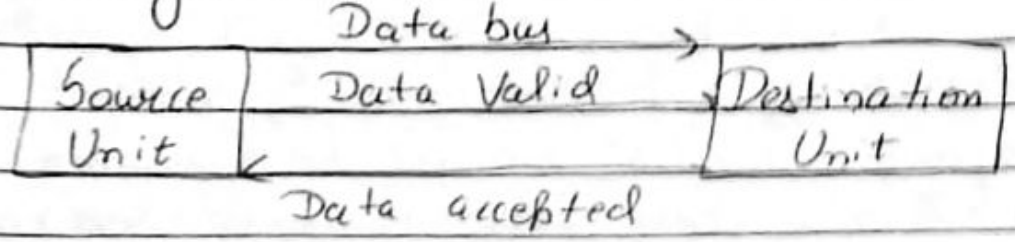
Tells when a valid data word is available in bus.

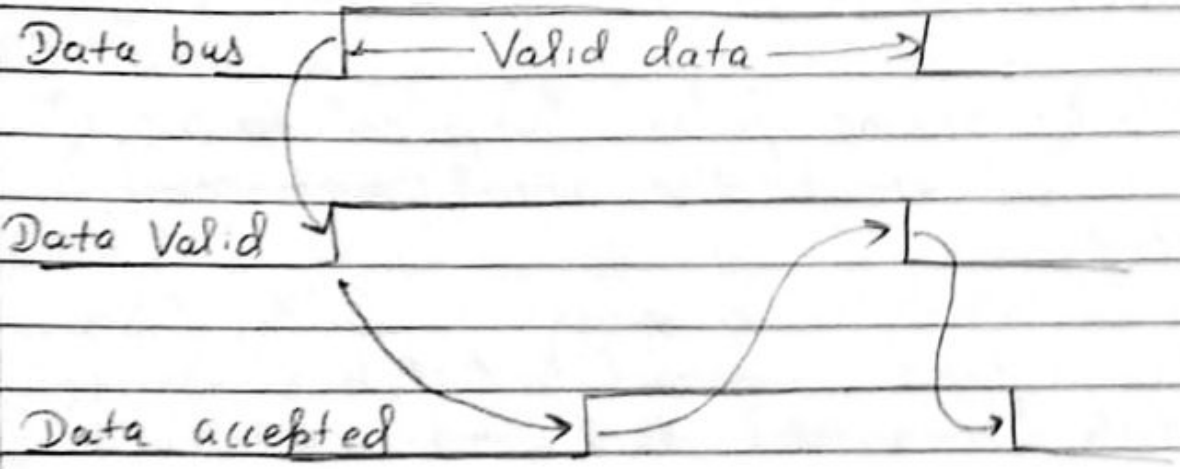


Source Unit first place data on data bus. After a brief delay ensure that data settle to steady value Source activate Strobe pulse. Now info on the data bus & Strobe signal remain steady in active state for a significant period to allow Dest to receive data.

Destination Unit use falling edge of Strobe to transfer data into internal reg. The Source removes data from bus after a brief period it disable its strobe pulse.

Handshaking:-





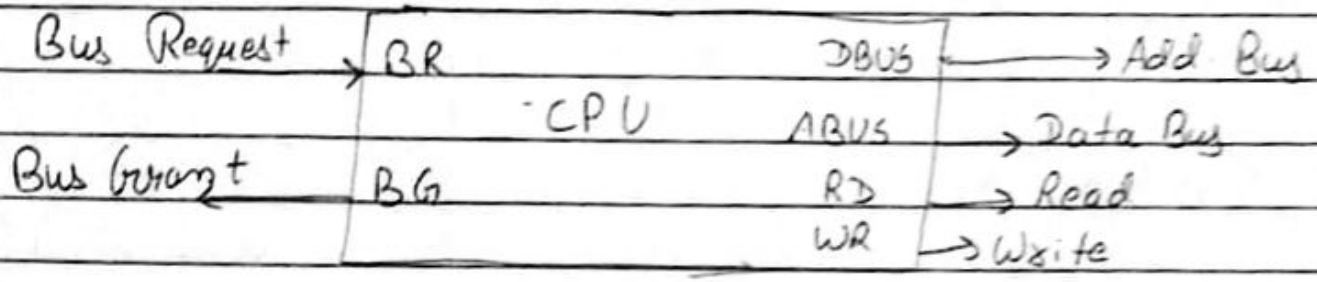
Data transfer procedure when initiated by Source. The two handshaking lines are data Valid, generated by Source & data accepted generated by destination Unit.

Source Unit initiates by placing data on bus & enable its data Valid signal. The data accepted signal is activated by destination after accepts the data. Source Unit then disables data Valid signal then destination disables its data accepted signal.

Direct Memory Access

Transfer of data b/w fast storage device like magnetic disk & memory is limited by CPU.

Removing CPU & let peripheral devices manage memory bus directly improve speed of transfer. This technique called (DMA).



DMA has two control signal BR & BG to serve the facilities.

Bus request:- Input used by DMA to request the CPU to relinquish control of buses. When input is active CPU terminates the execution of current instruction & place add bus, data, read write line in high impedance state

Bus Grant:- CPU activate BG to inform the DMA that buses are in high impedence state ∴ DMA now take control of buses to conduct memory transfer

When DMA terminates transfer & then CPU again take back control of buses.

Date: __/__/____

MON TUE WED THR FRI SAT SUN

Bus transfer

a) Brust transfer :- In DMA brust transfer a block sequence consisting of a no of memory words is transferred in a continuous brust will DMA controller is master of memory buses.

b) Cycle stealing :- In this DMA transfer one data word at a time & then give control back to CPU.

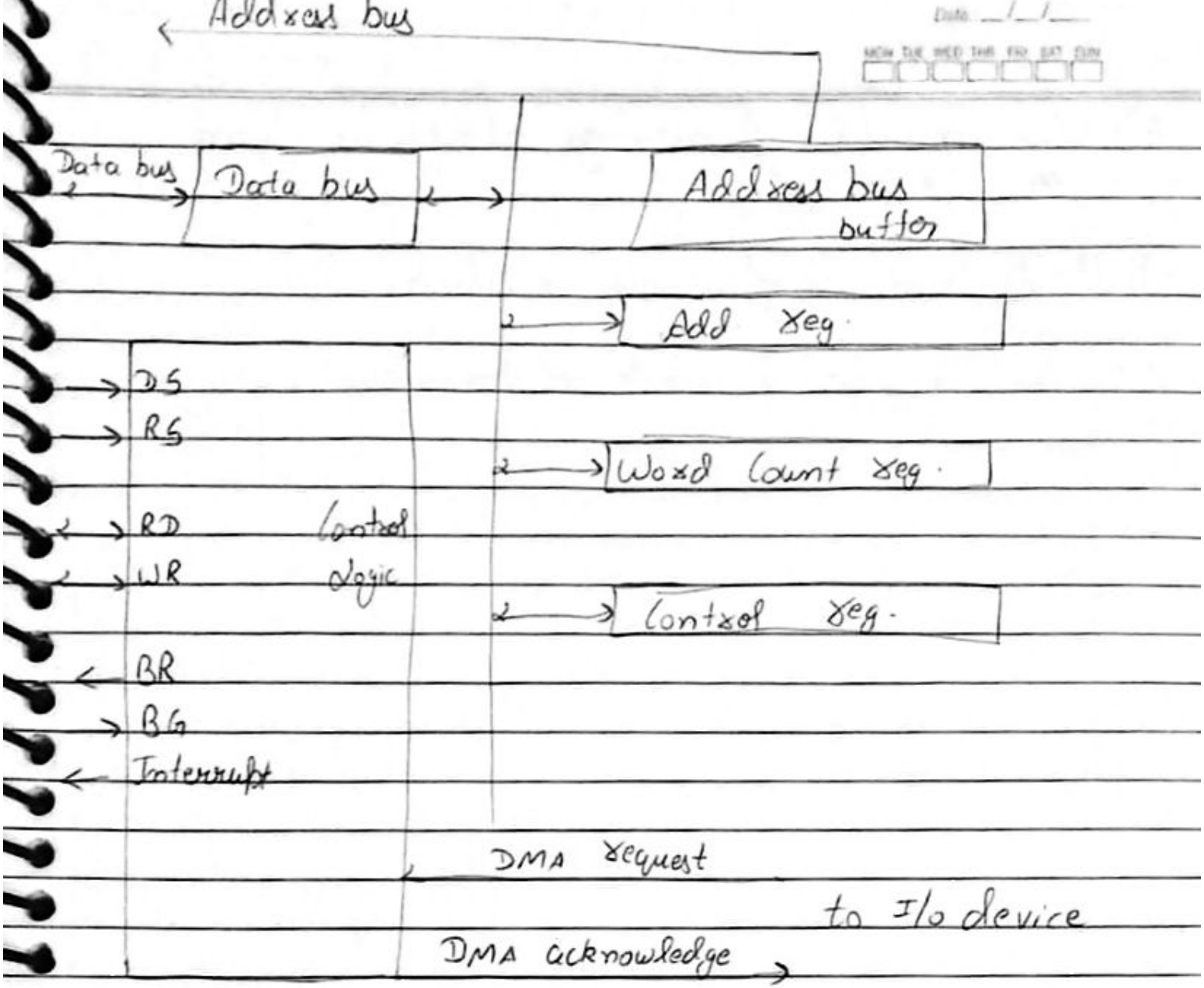
DMA Controller

→ Usual interface b/w CPU & I/o device. It needs an add reg., a word count reg, a set add. line

→ Registers in DMA are selected by the CPU through add. bus enabling DS & RS inputs.

→ When BG is 0 CPU can communicate with DMA reg. through its data.

→ when BG is 1 CPU has relinquish the buses & DMA can directly communicate with the memory by specifying add. in the address bus & activating RD & RW



Add reg:- Contains an address to specify desired location in memory. Incremented after each word transferred to memory.

Word Count reg:- Specifies no. of words must be transferred. This is ↓ by one after each word transfer.

Control reg:- Specifies mode of transfer CPU initialise DMA by sending

Date: __/__/____

MON TUE WED THR FRI SAT SUN

a) Starting add. of memory block for read or write

b) Word Count no of words in block

c) Control to specify mode of transfer.