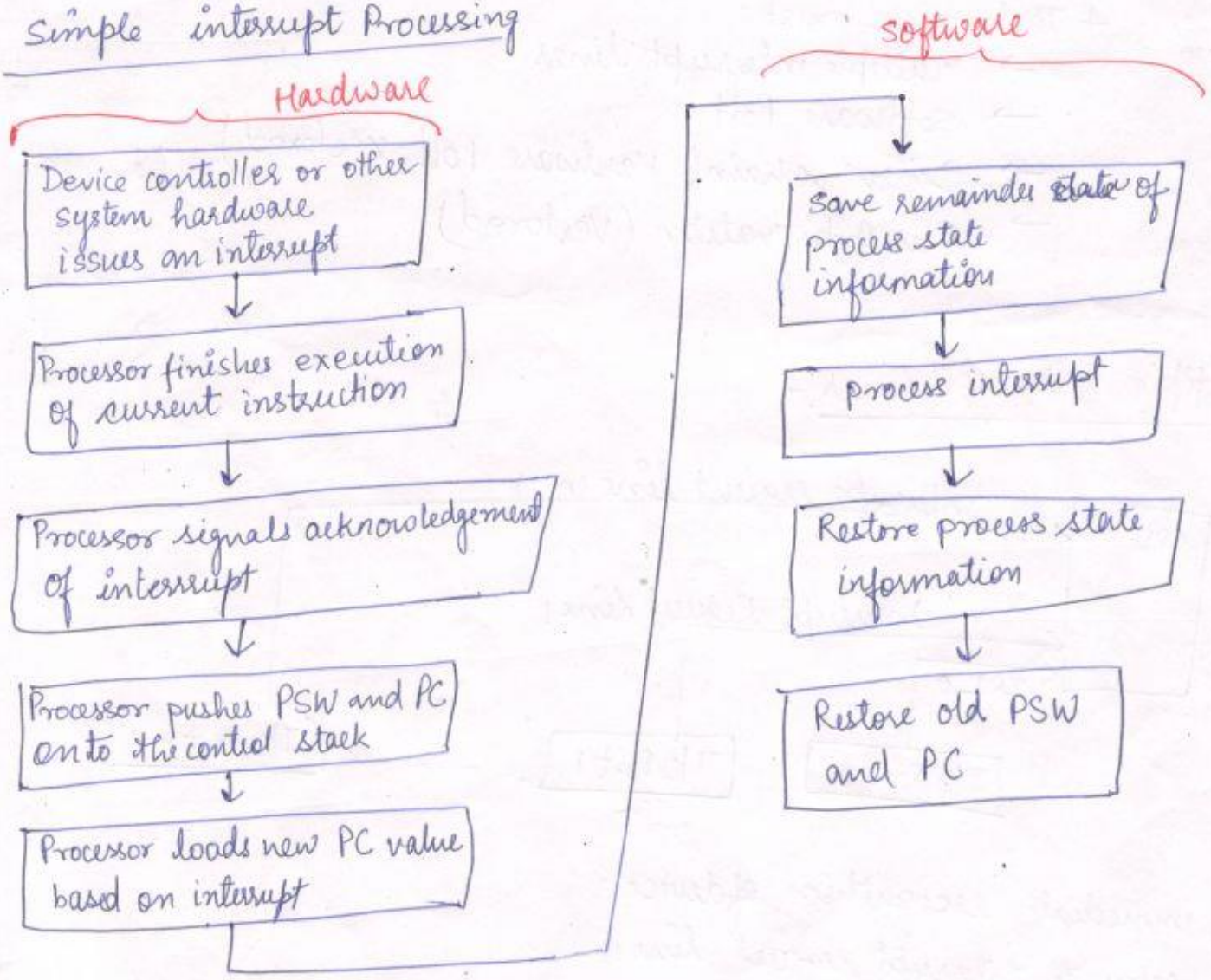
Interrupt Processing:

The basic method of interrupting the CPU is done by activating a control line that connects the interrupt source to the CPU.

CPU ⇒ * recognizes the presence of interrupt
* executes a specific interrupt handling program.
* determines the source of interrupt
* determines the address (branch address) of the interrupt handling program.

Simple interrupt Processing



Design issues:

Two design issues in implementing ~~I/O~~ Interrupt I/O.

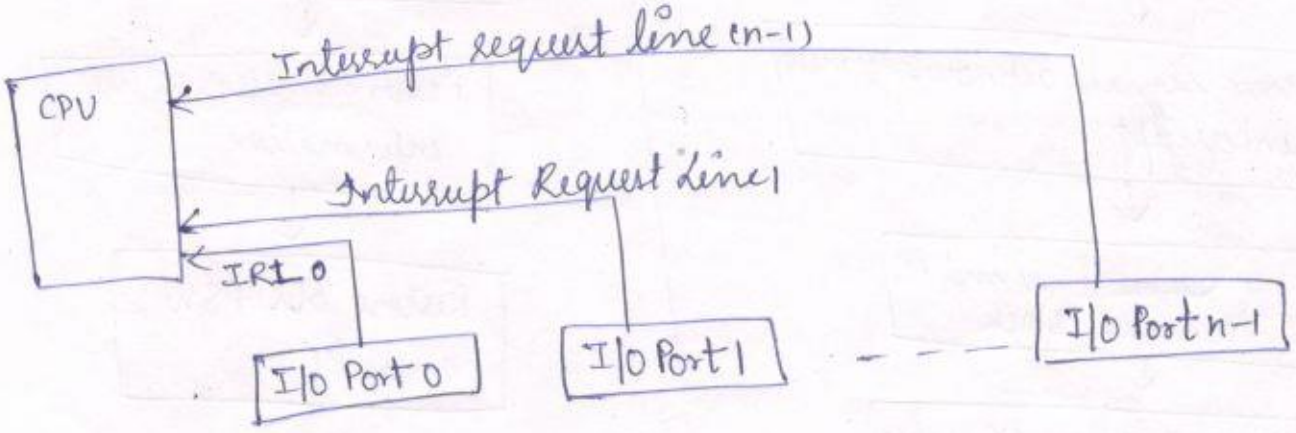
(1) → To determine which device issued the interrupt from multiple devices.

(2) if multiple devices interrupts have occurred, how does the the processor decide which one to process.

Issue: Device identification

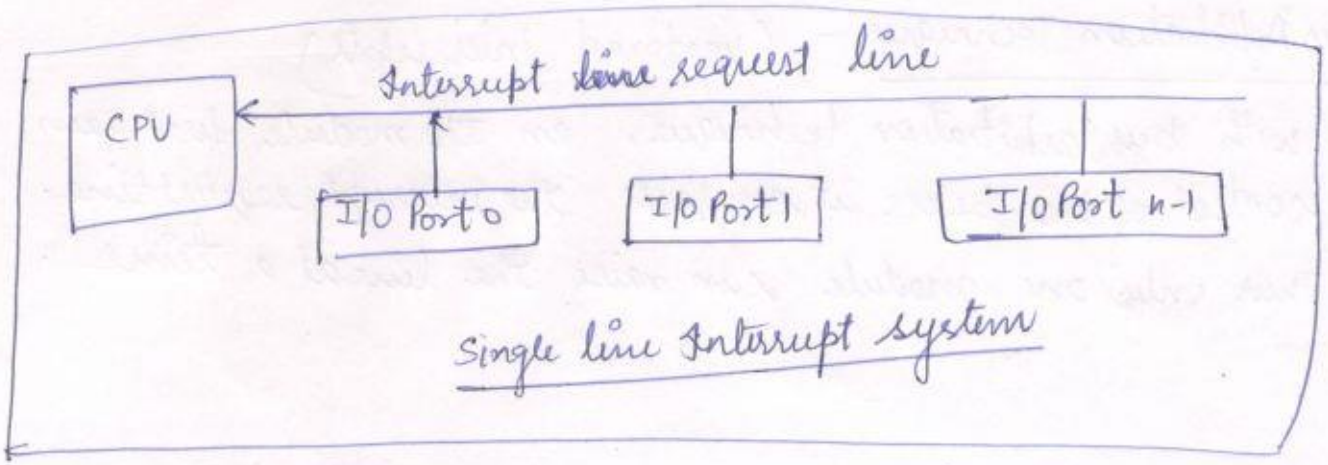
4 Techniques exist.
- → Multiple interrupt lines
- → Software Poll
- → Daisy chain (Hardware Poll, vectored)
- → Bus Arbitration (Vectored)

Multiple Interrupt Lines:-



⇒ immediate recognition of device.
⇒ separate Interrupt request line
⇒ impractical approach.

multiple Interrupt line system



Single line Interrupt system

Software Poll:

⇒ when processor detects the interrupt, it branches to an interrupt service-routine whose job is to poll each I/O module to determine which module caused the interrupt.

⇒ Time consuming

⇒ Priority can be implemented by defining polling sequence.

Vectored Interrupt using Daisy chaining: (Hardware Poll)

⇒ All I/O module shares common interrupt request line

→ The interrupt acknowledgement line is daisychained through the modules.

– when a processor senses the interrupt, it sends out an interrupt acknowledge propagating through a series of I/O module until it gets a requesting module.

– The requesting module responds by placing a word on data lines. This word is referred to as vector.

Bus Arbitration Techniques:- (vectored interrupts)

→ with bus arbitration technique, an I/O module first gain control of bus before it can raise the interrupt request line.

→ Thus only one module can raise the line at a time.

Types of Interrupts:

Program interrupts

These are generated by some condition that occurs as a result of an instruction execution such as:

- → Arithmetic Overflow
- → Division by zero
- → Execution of illegal machine instruction
- → Segment limit violation
- → Execution of privileged instruction

Amit

Timer Interrupts

These are generated within the processor. This allows operating system to perform certain operations on regular basis.

Input-output interrupts:

These are generated for initiation or completion of I/O operations I/O failure or I/O error too can generate an interrupt.

Hardware Failure Interrupts

These are generated by a failure; such as power failure or memory parity error.

Interrupts vs. Exceptions:

An interrupt is generated by a signal from hardware, and it may occur at random times during execution of a program.

An exception is generated from software, and it is provoked by the execution of an instruction.

Hardware and Software interrupts

→ When microprocessor receive interrupts signals through pins (hardware) of microprocessor. These are known as Hardware interrupts

→ Software interrupts are those which are inserted in the between the program which means these are mnemonics of the microprocessor

Vectored and non vectored interrupts

- In vectored interrupts, the source (or device) that interrupts supplies the branch information to the computer.
- In a non-vectored interrupt, the branch address is assigned to a fixed location in the memory.

Maskable and non-maskable:

- Maskable interrupts are those which can be disabled or ignored by the microprocessor
- Non-maskable: which can not be disabled or ignored by the microprocessor.

Types of interrupts:

4 categories
- → Program interrupts
- → Timer interrupts
- → Input/output interrupts
- → Hardware failure

Direct memory Access :-

→ Both interrupt driven and programmed I/O require involvement of CPU for data transfer.

→ CPU can be better utilized for program execution.

→ I/O activities are slow and involvement of CPU will not have a desired effect on the system performance.

→ DMA increases the speed of I/O transfer by eliminating the role played by CPU in I/O operations.

→ when large amount of data is stored/transferred from CPU, a DMA module can be used.

DMA operates in the following ways:

when I/O is requested, the CPU instructs the DMA module about the operation, by providing information:

- → Source address
- → Target address
- → Read / write
- → Number of words to be read or written.

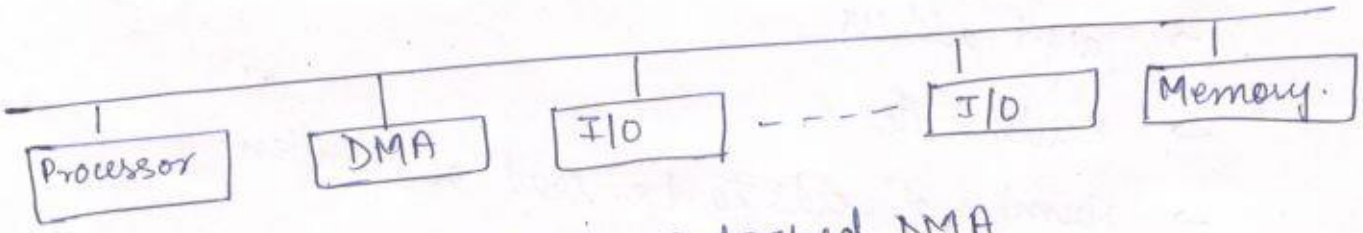
⇨ CPU loads the DMA registers 'Data count' and 'Address Register' with number of bytes to be transferred and starting address memory location respectively.

When the DMA controller is ready to transmit or receive data, it activates the DMA request line to CPU. CPU wait for the next break point. CPU now leaves the control of system bus and activates DMA acknowledge.

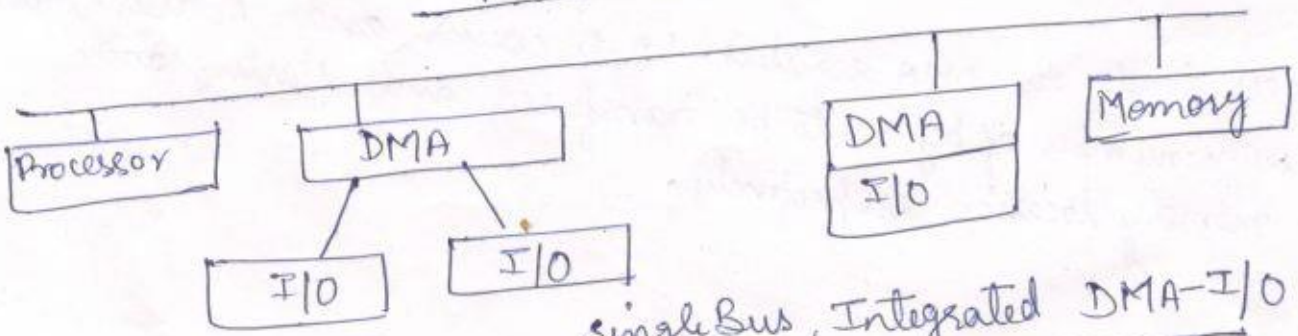
→ The DMA controller transfers the data directly from or to main memory. After, a word is transferred. Data count (or word count register) and Address register are updated.

→ If the data count register has not reached zero but the I/O device is not ready to send or receive the data, the DMA controller releases the system bus to CPU by deactivating DMA request line.

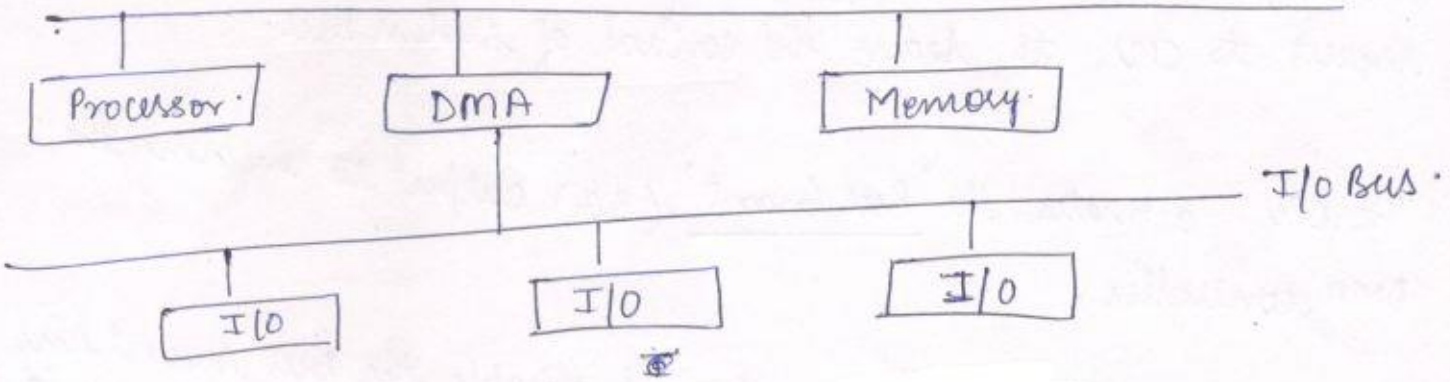
- if data count register is decremented to zero, DMA controller finally relinquishs the control of system bus. It may also send an interrupt signal to CPU to indicate the end of data transfer.



Single Bus Detached DMA

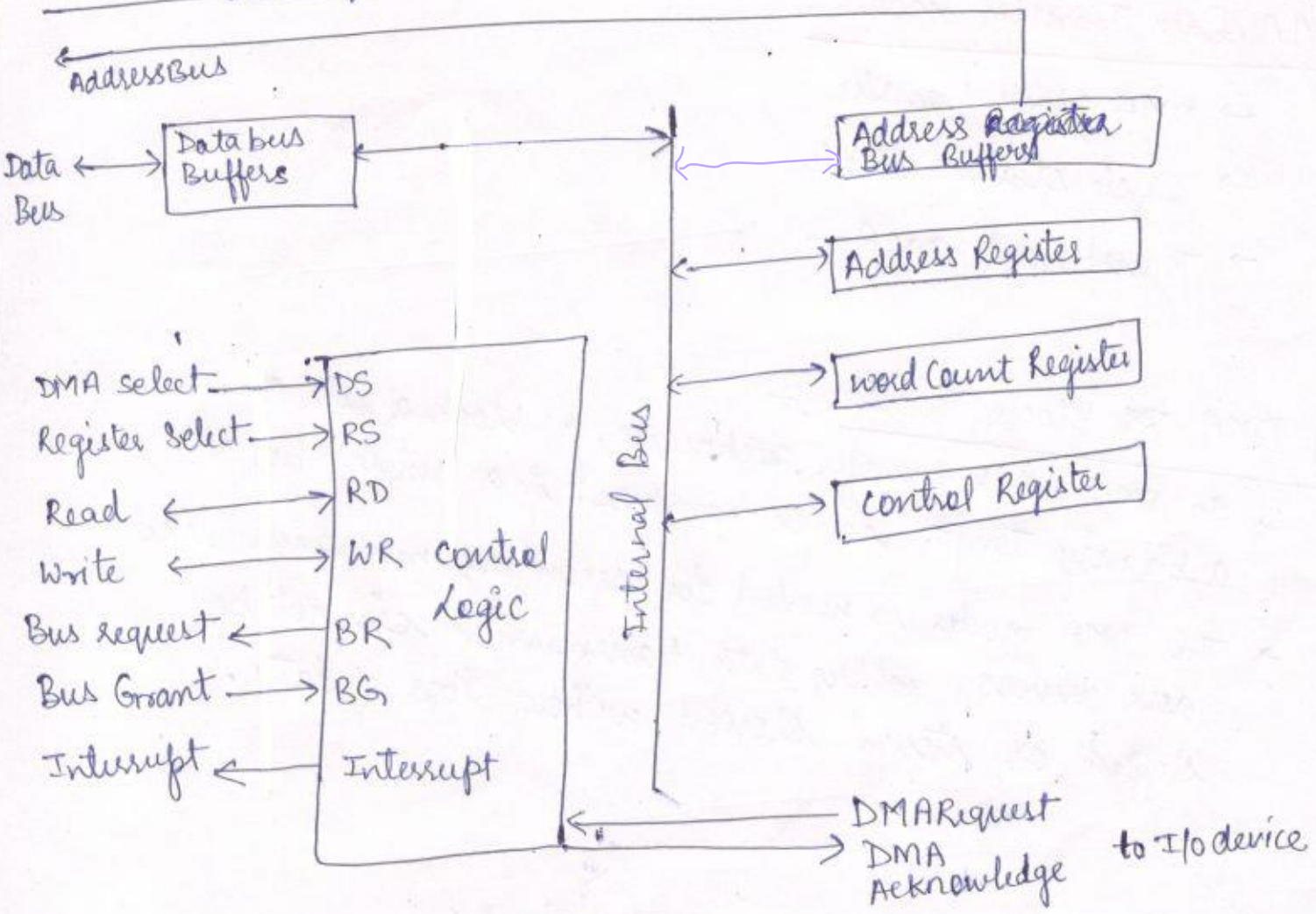


Single Bus, Integrated DMA-I/O



I/O Bus

Block Diagram of DMA



→ Bus Request (BR) input is used ~~to~~ by the DMA controller to request to CPU to leave the control of system bus.

→ The CPU activates the 'Bus Grant' (BG) output to inform the DMA controller.

→ When DMA terminates the transfer, it disable the bus request line. CPU disables the bus grant, takes control of buses and returns it to its normal operation.

DMA Data Transfer modes:

→ DMA block transfer
→ Cycle stealing mode
→ Transparent DMA.

(1) DMA ~~B~~ Block Transfer:-

→ In DMA block transfer technique, a block of data of arbitrary length can be transfered in a single burst.
→ This DMA mode is needed for secondary memories like disk drivers, where data transmission can not be stopped or ~~stowed~~ slowed without loss of data.

② cycle steading Mode:

In cycle stealing mode, the DMA controller is allowed to transfer one word at a time, after which it must return the control of the bus to the CPU.

* DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.

Note that this is not an interrupt; the processor does not save a context and do something else.
DMA module transfers a word and returns the control to the processor Rather, processor pauses for one bus cycle. The overall effect is to cause the processor to execute more slowly. Nevertheless, for a multiple-word I/O transfer, DMA is far more efficient than interrupt driven or programmed I/O.

③ Transparent DMA:

In transparent DMA, DMA is allowed to steal only those cycles when the CPU is not using the system bus.
CPU does not require to have control of system bus during Decode instruction or execute instruction phase.

* DMA transfering data during transparent DMA does not have any adverse effect on CPU performance.