Varun Khadayate A016 02-11-20

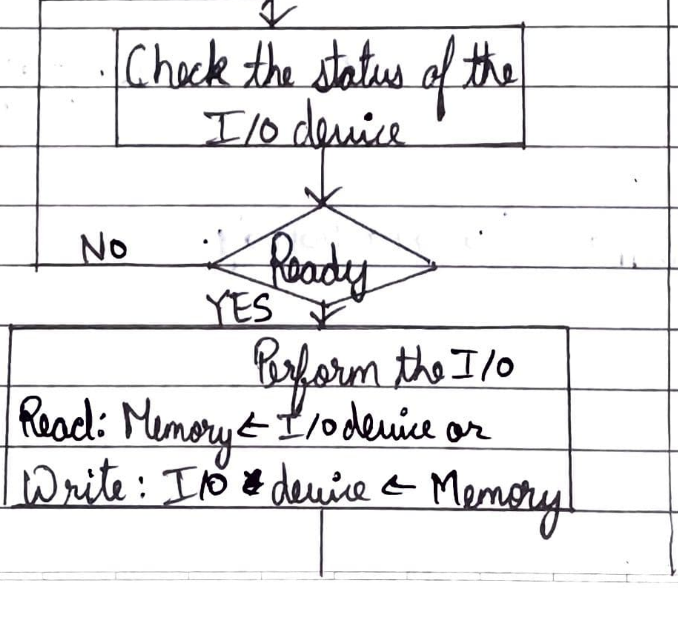
COA

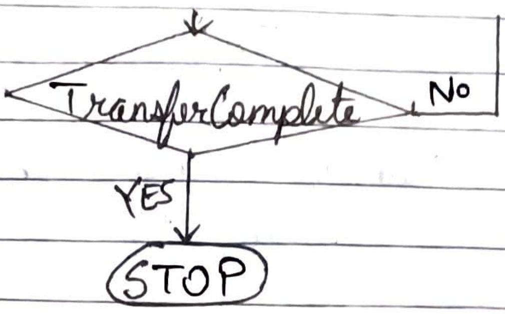
Assignment -III

Q1. Programmed I/O

- Programmed I/O is the simplest techniques of performing input/output data transfer and it is very effective when the number of device is less in system.
- It is best on the concept of the program controlling the input and output, data transfer.

1. CPU generates the desired I/O command.
2. CPU checks the status of I/O device whether it is ready to perform the specific read or write operation.
3. If device is not ready CPU remains "BusyWaiting".
4. If device is ready, the specified Read or Write I/O operation is completed.
5. It is checked whether any more I/O transfers are pending.

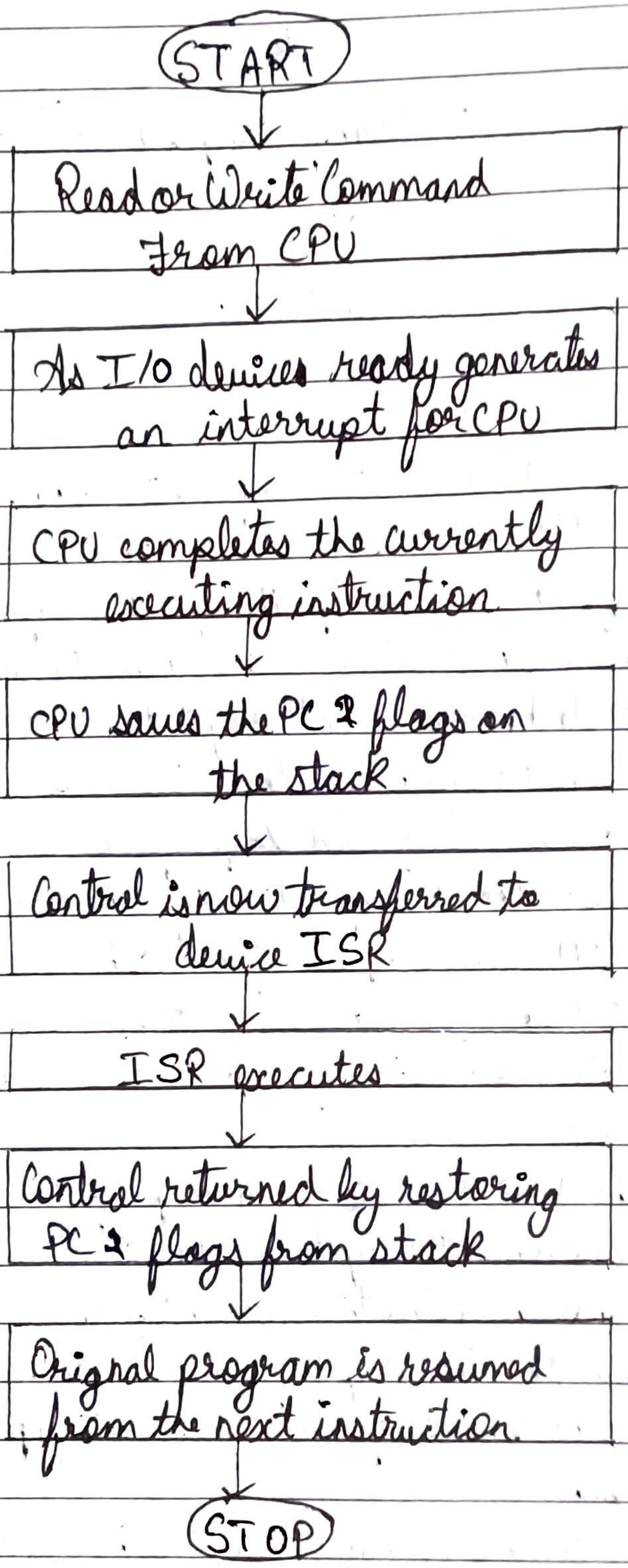

START
Read Or Write Command From CPU
Check the status of the I/O device
Ready
NO
YES
Perform the I/O
Read: Memory ← I/O device or
Write: I/O device ← Memory



Interrupt Driven· I/O

- Programmed I/O is not an efficient technique due to the fact that the CPU consumes substantial amount of line in checking the device repeatedly turn by turn.
- Furthermore, the devices are also have to wait for their turn. The better technique is interrupt driven I/O. In this method, the CPU is not sequencing the I/O device on its own.
- When the device is required to perform I/O data transfers its intimates so the CPU by activating the hardware line which goes to CPU as an input signal. This line is called as Interrupt ~~+~~ therefore this I/O is termed as Interrupt Driven I/O

1. CPU generates instructions for Read or Write operation from/to I/O device.
2. When the I/O device is ready to perform I/O data transfer, it generates an interrupt to the CPU.
3. CPU completes the instruction currently in execution and then recognizes the interrupt from I/O device.
4. It saves the context of currently executing problem on the top of the stack.
5. Then the control is transferred to the start of Interrupt Service Routine [ISR]
6. Now the CPU services the Interrupting I/O device performing the underlying I/O data transfer.
7. Then the CPU comes back, retrieving the context from the stack and continues the orignal ~~se~~ execution executing ~~problem~~ program resuming it from the next instruction onwards.


START
Read or Write Command From CPU
As I/O devices ready generates an interrupt for CPU
CPU completes the currently executing instruction
CPU saves the PC & flags on the stack.
Control is now transferred to device ISR
ISR executes
Control returned by restoring PC & flags from stack
Orignal program is resumed from the next instruction.
STOP

Q3. Priviledged Instruction

The that can run only in Kernel Mode are called as Priviledged Instructions.

Characteristics:

- If any attempt is made to execute a Priviledged Instruction in User Mode, then it will not be executed and treated as an illegal instruction.
- Before transferring the control to any User program, it is the responsibility of the OS to ensure the Timer is set to Interrupt.
- Priviledged instruction are used by OS in order to achieve correct operation.
- Eg: I/O instruction & Halt Instruction, Set the Timer.

Non-Priviledged Instruction

The instructions that can run only in User Mode are called Non-Priviledged instruction.

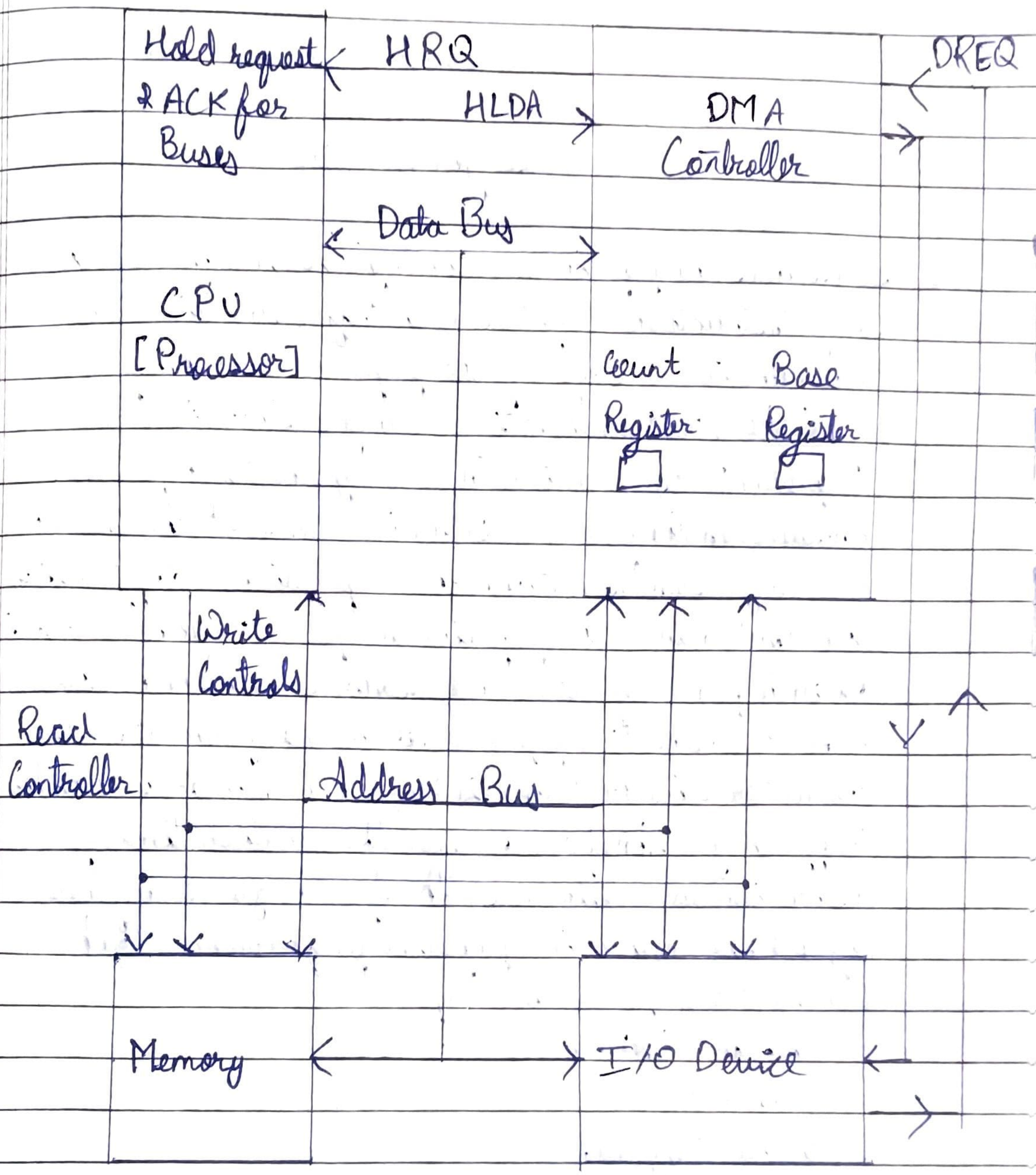
Eg: Reading the status of Processor, Reading System Time etc.

Q4. Exceptionsare unexpected events which will disrupt the normal flow of execution of instruction. ~~An exception is an~~ Whenever an exception occurs, the hardware starts executing the code that performs an action in response to the exception. The action may involve killing a process, outputting an error message or communicating an external device. The instruction responsible for this action reside in OS Kernel called as interrupt handler code. After the handler code is executed, it may be possible to continue execution after the instruction where the execution occured.

Q2. The technique of accessing memory directly without the involvement of CPU for performing I/O data transfer, is called Direct Memory Access or DMA. The data transfer involving DMA technique are the transfers from the memory to I/O device, I/O device to memory & from one memory to another memory. These data transfers can be performed without direct involvement of the CPU. However, they do require the data bus, address bus and control bus. And, by default they all are in the control of the CPU.

The sequence of steps for performing I/O data transfer using DMA controller are:

1. CPU programs the internal register of DMA controller & prepares it for DMA activity.
2. When the Device is Ready for DMA transfer, it activates DREQ signal to the DMA controller. In turn, DMA controller activates hold signal to the CPU.
3. CPU completes the current machine cycle & then tri-states the system buses & stop driving them.
4. DMA controller takes over the control of buses. It issues appropriate addresses & control signal to perform DMA based data transfer.
5. As DMA activity is over, device deactivates DREQ signal to DMA controller & in turn gets #DACK signal to be deactivated.
6. DMA controller stops driving the buses & deactivates HOLD signal to the CPU
7. CPU takes back the control of bus & indicates so by deactivating HLDA signal. Subsequently, CPU continues its work normally.



Hold request
& ACK for
Buses
HRQ
HLDA
DREQ
DMA
Controller
Data Bus
CPU
[Processor]
Count
Register
Base
Register
Write
Controls
Read
Controller
Address Bus
Memory
I/O Device