EE2016 Microprocessor Theory & Lab
Fall, 2024

Week8: Polling, Interrupts & DMA Comparison Problem

Dr. R. Manivasakan. OWSM Lab, EED, IIT, Madras

Interfacing Peripherals with CPU

- Recap
  - Till now
    - From FF to memory block
      - Memory block representation with data, address and control buses
      - Memory along with ALU
    - Execution Unit
      - Internal registers, status registers, multiplier, shifter etc, along with ALU formed EU. Each sub-engines studied in detail
      - Studied EU along with the first instruction MOV, then LDI
      - Studied many example assembly programs
    - Control Unit
      - PC, Stack, SP, Instruction Decoder, hardwired control unit implementation
      - Studied in detail, each sub-engines
  - Together EU with CU formed microprocessor
    - Studied interfacing peripherals through MMIO (Exp 8)
      - Step 1 Addressing
      - Step 2 Handshake: Polling, Interrupt & DMA
- What next?
  - Comparison

We saw that ...

- Polling registers in Peripheral Controller used to detect state of Peripherals
  - Would require frequent polling and waste of CPU cycles
- Interrupt is an alternative means to draw attention to an external event occurring at a peripheral
  - Enables CPU to suspend its current task and handle external event through Interrupt subroutine (ISR) which reads status / data registers of a peripheral controller
  - CPU would resume suspended task after the ISR exits, without any glitch
  - Nested Interrupts and Interrupt Priority used to handle interrupts form multiple external events
- Interrupt Controller chip (like a Peripheral Controller) sometime used to connect multiple interrupts, define priorities for each interrupt and turn on / off any Interrupt

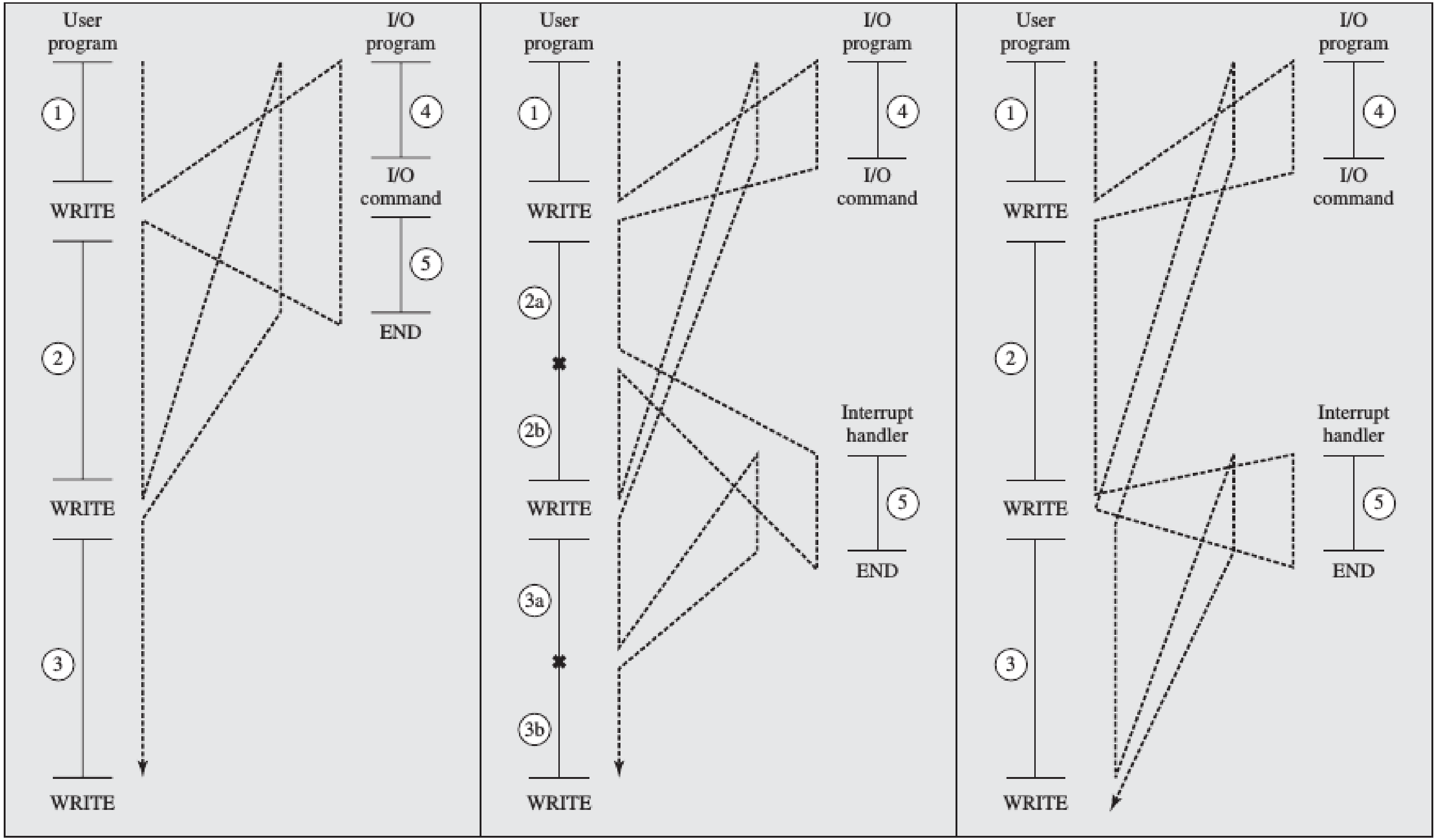
Advantages and Disadvantages of DMA

- Advantages
  - DMA allows a peripheral device to read from/write to memory without going through the CPU
  - DMA allows for faster processing since the processor can be working on something else while the peripheral can be populating memory
- Disadvantages
  - DMA transfer requires a DMA controller to carry out the operation, hence cost of the system increases
  - Cache Coherence problems: to be discussed later

Interrupt versus DMA

S. No	Issue	Interrupt	DMA
1	Principle of operation	The input/ output module / peripheral proactively interrupts the microprocessor, to request service, whenever need arises. The microprocessor or microcontroller has to be confgured accordingly for honoring the interrupts.	In Direct Memory Access (DMA), the CPU grants I/O module authority to read from or write to memory without involvement.
2	Time and power consumed	Interrupt-driven input/ output still consumes relatively considerable time and power because all data has to pass through processor.	DMA consumes less power and less time for a given task of memory transfer as the number of entities are just two: memory and the peripheral (or two memory modules)
3	I/O transfer rate	The I/O transfer rate is limited by the speed with which the processor can test and service a device.	The I/O transfer rate is limited by the speed set by the technologies of the memory and the peripherals.
4	CPU activity	The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.	CPU can do its own job but using only internal bus (as system bus is used by DMA).
5	Amount of data transfer	Smaller amount of transfer, interrupt is not a bad scheme	The very advantage of DMA is reaped when the volume of data transfer is large
6	Special case where CPU is slow	I/O data transfer rate happens at CPU rate	Data transfer can happen at rate higher than that can handled by CPU

Primitive handshake versus Interrupt



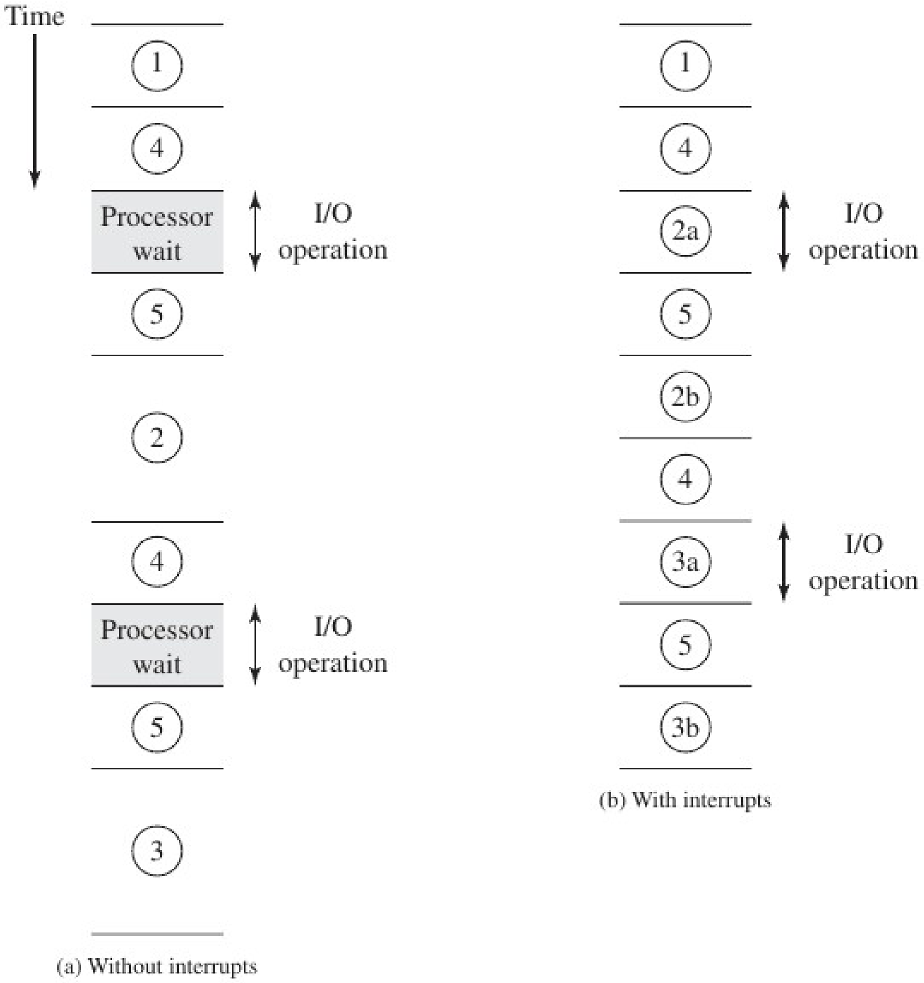
(a) No interrupts

(b) Interrupts; short I/O wait

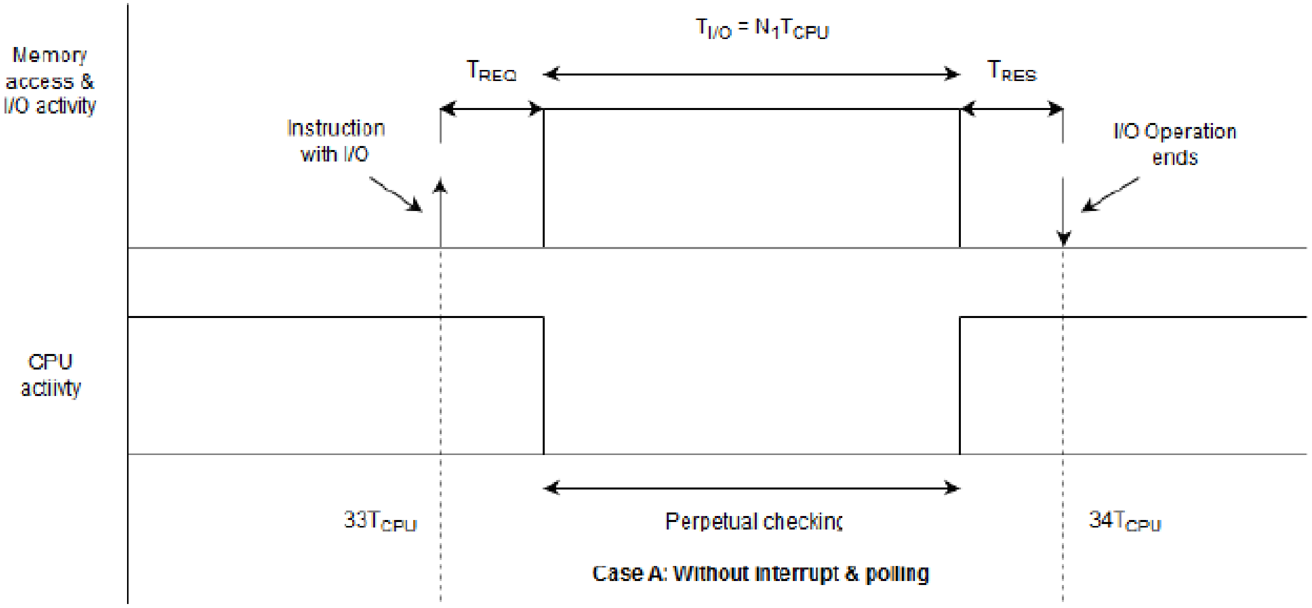
(c) Interrupts; long I/O wait

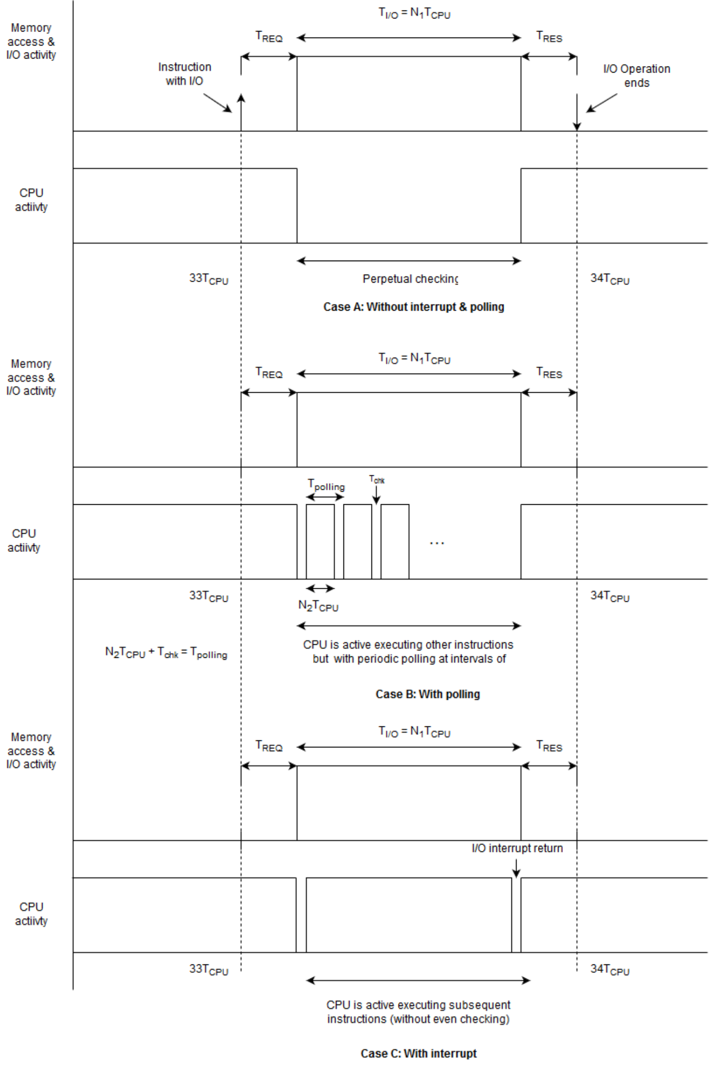
CPU behaviour under I/O Interrupt with Short I/O

- Without Interrupts
  - Once the I/O request is made, the CPU has to attend to it & has to wait till it is completed.
  - CPU cant do anything else
- With Interrupts
  - In tending to I/O requests, the CPU can assign the job & simultaneously could do other pending jobs
  - During requests and job completion instants the peripheral is the one which intimates the CPU.
  - Short I/O waits
  - Long I/O waits

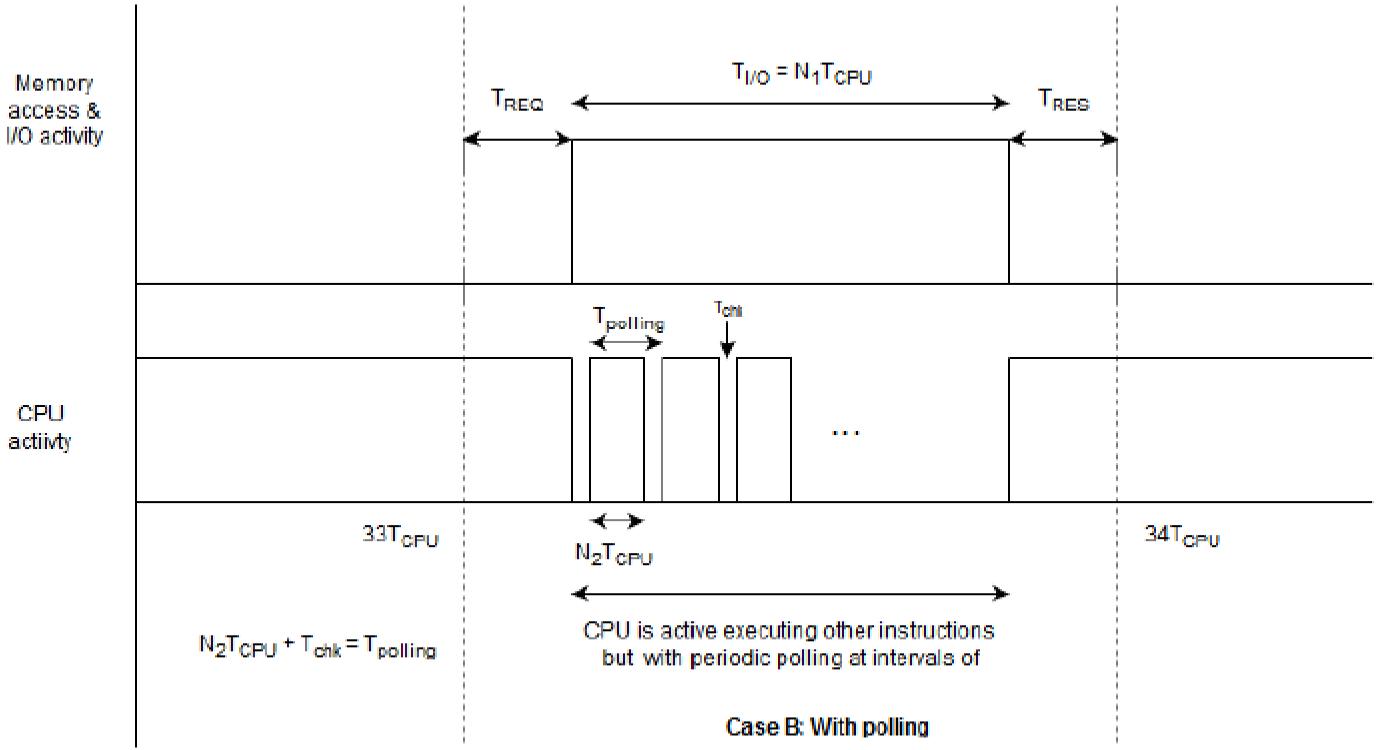


Evolution of handshaking into DMA

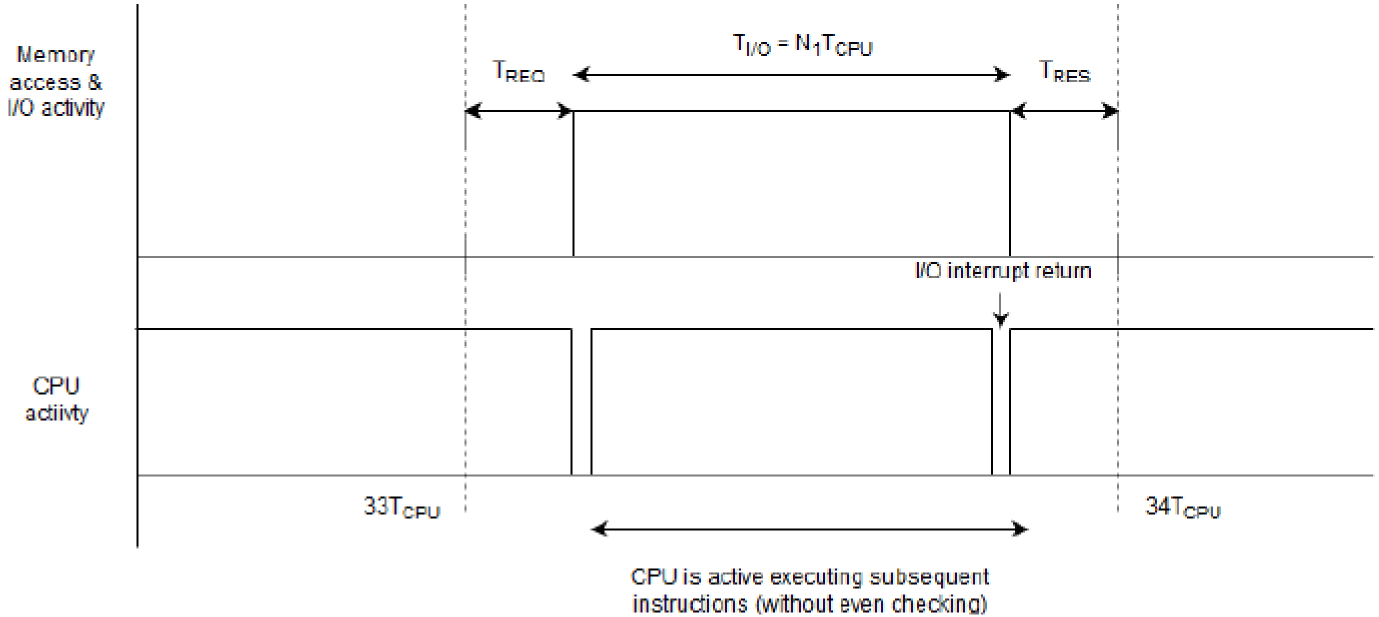




Evolution of handshaking into DMA


Memory access & I/O activity
T_{REQ}
$T_{I/O} = N_1T_{CPU}$
T_{RES}
CPU actiivty
$T_{polling}$
T_{chk}
...
$33T_{CPU}$
N_2T_{CPU}
$34T_{CPU}$
$N_2T_{CPU} + T_{chk} = T_{polling}$
CPU is active executing other instructions but with periodic polling at intervals of
Case B: With polling

Evolution of handshaking into DMA


Memory access & I/O activity
$T_{I/O} = N_1T_{CPU}$
T_{REQ}
T_{RES}
I/O interrupt return
CPU actiivty
$33T_{CPU}$
$34T_{CPU}$
CPU is active executing subsequent instructions (without even checking)

Case C: With interrupt