Cx1106
Computer Organization and Architecture

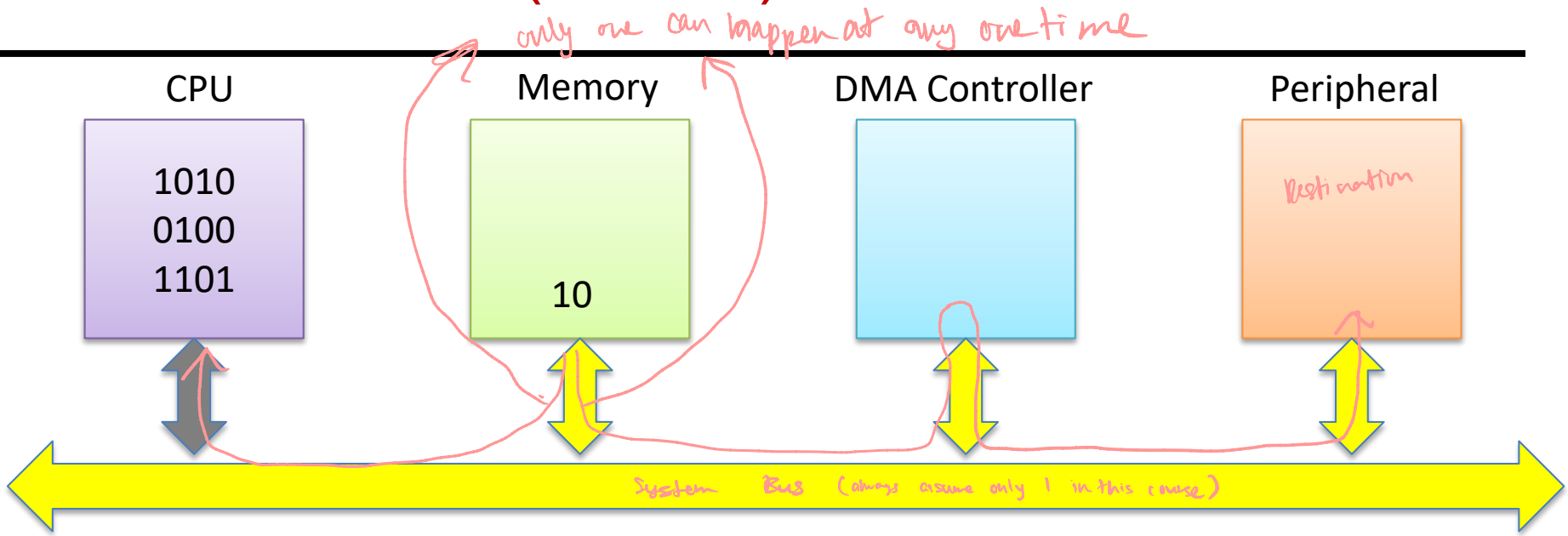
Direct Memory Access

Oh Hong Lye
Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

Optimizing CPU resources in Data Transfer

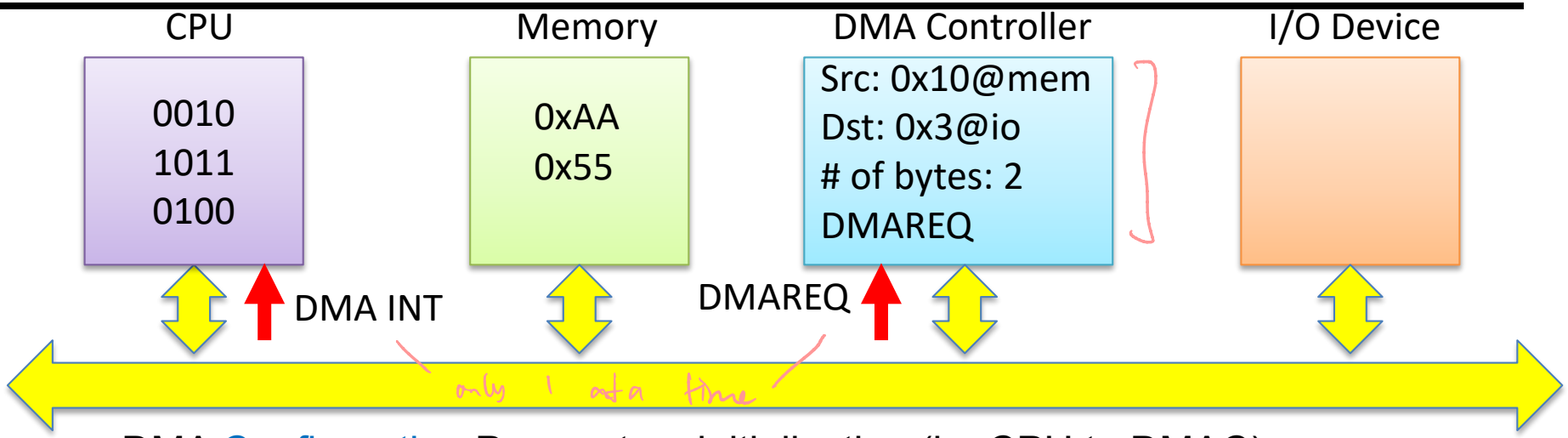
- Till now, we have been using CPU to perform the data transfer.
- This becomes increasing inefficient as amount of data increase as CPU has to spend most of its time moving data.
- As such, CPU has less time to perform algorithm processing.
- DMA controller (DMAC) is added to relief CPU of the data transfer task.
- DMAC has dedicated hardware that could move data more efficiently than CPU in scenarios where complex address manipulation is required. E.g. de-interleaving Left and Right Channel Audio data.
- If there are no conflict in hardware resources used, it is possible for data transfer via DMA and CPU execution to occur simultaneously.
- If there is a conflict in hardware resources used, e.g. both DMAC and the CPU needs the system bus, then access will be given to the one with higher priority.
- Who has the higher priority and whether the priority is configurable depends on processor design and is thus processor specific.

DMA Controller (DMAC)



- DMAC is a Data Bus Controller module that performs data transfer independent of CPU, between Memory and memory, I/O Peripheral and I/O peripheral, Memory and I/O Peripheral.
- Generates address and initiates read/write operation between devices mentioned above.
- Note that 'Peripheral' here could be internal or external peripherals
- The DMAC shown here uses the Fetch and Deposit DMA mechanism where the data goes into the internal buffer of the DMAC before being transferred to the Destination.

Basic DMA Process



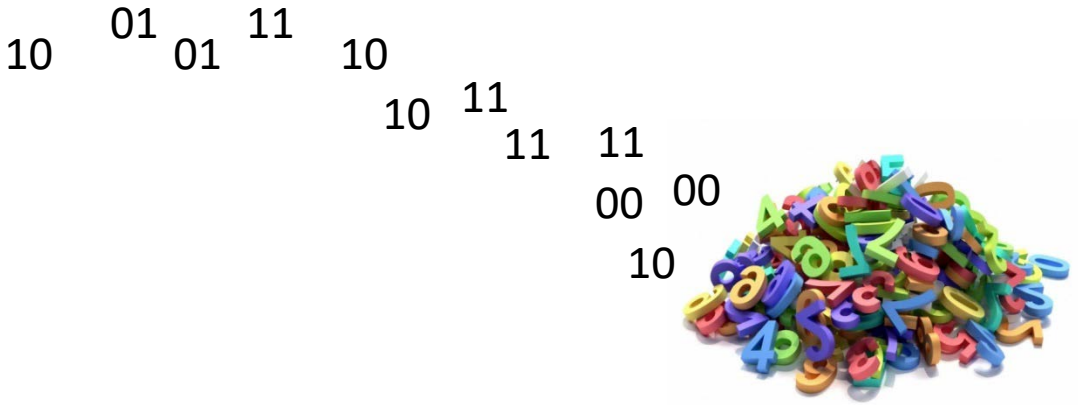
- DMA Configuration Parameters initialization (by CPU to DMAC)
  - Source Address
  - Destination Address
  - Amount of data to transfer
  - DMA Trigger signal (DMAREQ, interrupt, software bit etc)
- CPU proceed with its own task while DMAC wait for DMA Trigger Signal.
- Once DMA Trigger occurs, DMAC request to take over system bus.
- DMAC transfer data according to configuration.
- DMAC notify CPU of data completion (typically via interrupts) and release system bus.

DMA Mode of Operation

- Different processors has different DMAC design, typically with slight variance in terms of how they transfer data and the data type (Audio, Video etc) they are optimized for.
- In general, DMAC could transfer the block of data they are tasked to transfer in the following mode
  - Burst
  - Cycle Stealing
  - Transparent
- Do note that you may encounter DMAC design that varies from the basic mode of operation discussed here but general concept will still apply.
- In fact, majority of the time, you'll encounter DMAC design that uses a combination of the modes described

Burst Mode

- The DMA controller gains control of the data bus and transfer multiple units of data before returning control of the bus to the CPU.
- Note that the burst could be the entire block of data DMAC is tasked to transfer, or a subset of the block, i.e. it may take a few burst to complete the transfer of the entire block.
- CPU may continue to operate as long as it does not need access the particular data bus that DMAC is using.
- If CPU needs to access the data bus, CPU may be suspended till DMAC has completed its data transfer.
- Fast data transfer rate but may render the CPU inactive for longer period of time.



Cycle Stealing

- DMAC releases the data bus after transferring one unit of data.
- Depending on processor design, DMAC tends to execute the data transfer
  - Between CPU instructions
  - Between Pipeline stages
- CPU may be suspended if it need to access the data bus but suspend time is shorter as only one unit is transferred at one time.
- Transfer rate is slower than in Burst Mode but will CPU will only be inactive for very short period of time.
- Favoured in application which requires CPU to be responsive e.g. real-time security status monitoring.

Between CPU Instr

CPU	Instr1	Instr2		Instr3		Instr4
DMAC			Data		Data	

Between Pipeline Stages

CPU	Fetch Instr	Decode Instr		Fetch Operand	Exe Instr	
DMAC			Data			Data

Transparent Mode

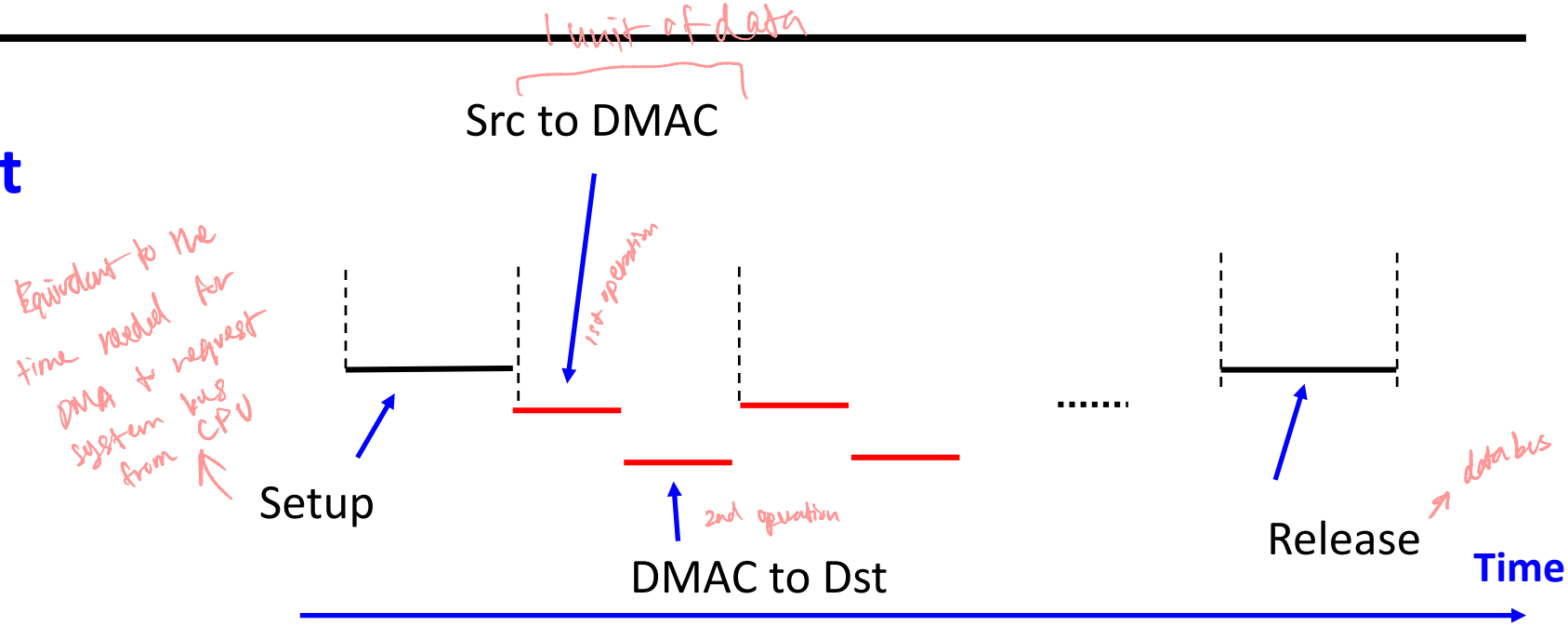
- DMAC transfer data only when CPU is not using the data bus.
- Zero impact to CPU performance in terms of data bus access.
- Potentially the slowest transfer rate among the three modes.
- More Complex hardware needed to detect when CPU is not using the data bus. → before DMA can start its transfer
- The CPU activity detection technique could also be used by some processors as a cue to trigger a burst transfer only when CPU is not using the system bus.

Comparison of DMA Transfer Modes

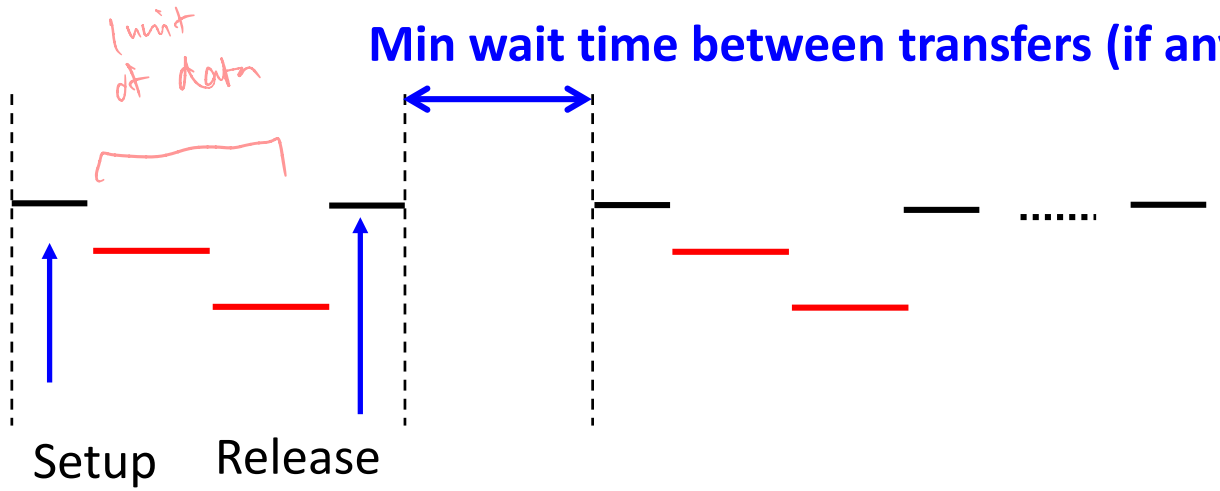
- **Burst**
  - Allow faster data transfer
  - CPU may be suspended for longer period of time
  - May not be suitable for Real-time application ↗ depending on how long
  - Suitable for application where transfer is bursty e.g. HDD file transfer.
- **Cycle Stealing**
  - Interleaving DMA data transfer with CPU instructions allow CPU to continue executing its program while DMAC is doing the data transfer.
  - CPU performance lower due to interleaving of DMA transfers, but would still be more responsive than in Burst mode.
  - Slower data transfer rate than Burst mode.
  - Suitable for Real-time application.
- **Transparent**
  - Potentially would not affect CPU performance at all.
  - Slowest data transfer rate but best CPU respond
  - More complex hardware needed to detect when CPU is not using the bus.

Fetch and Deposit (Timings)

Burst



Cycle Stealing

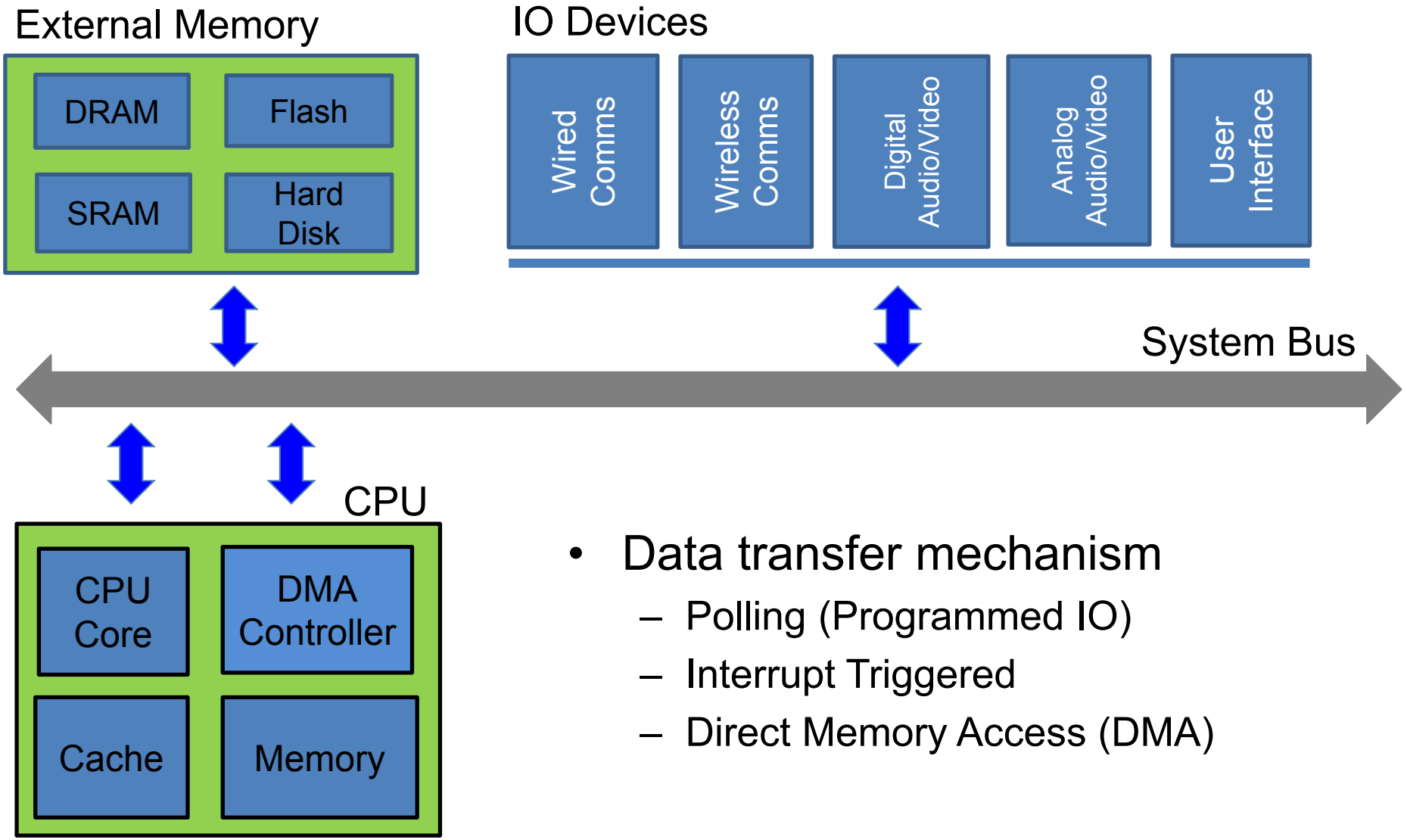


Cx1106
Computer Organization and Architecture

Data Transfer Mechanism

Oh Hong Lye
Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

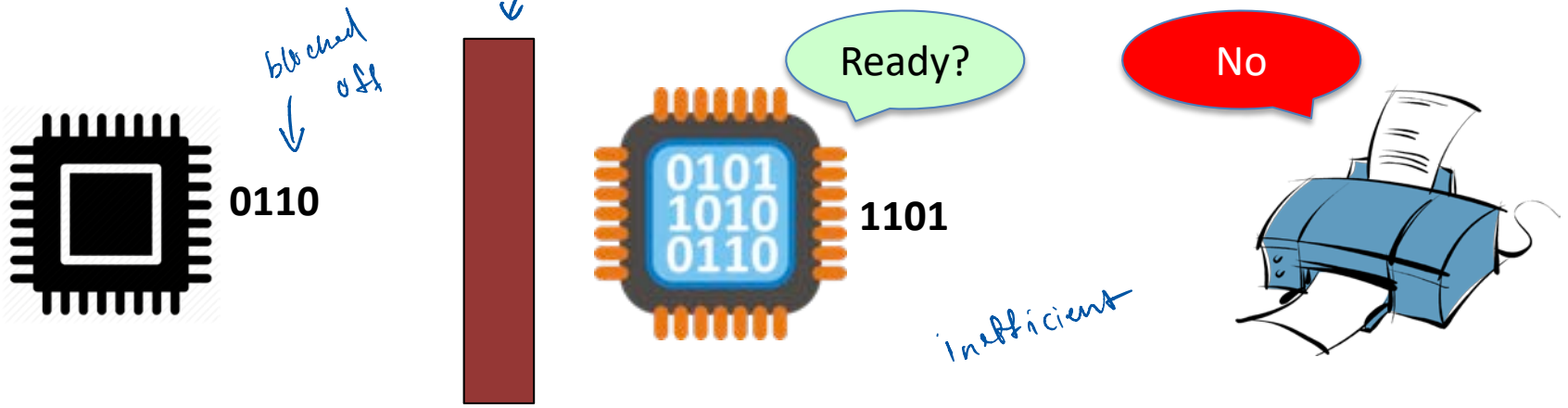
Data Transfer Mechanism



- Data transfer mechanism
  - Polling (Programmed IO)
  - Interrupt Triggered
  - Direct Memory Access (DMA)

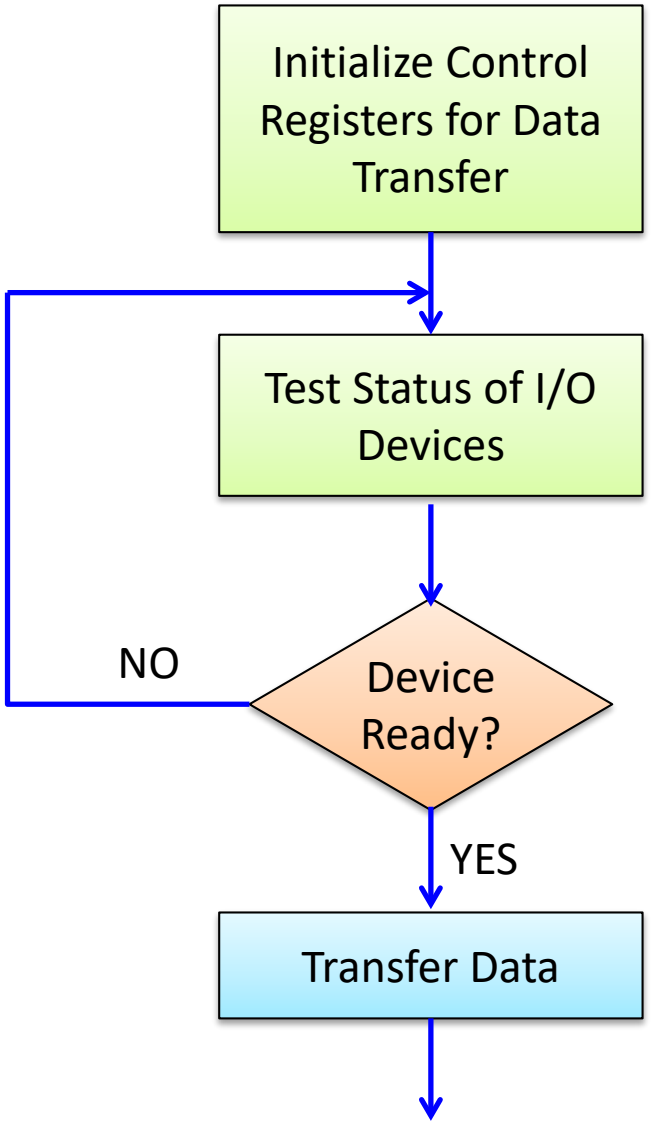
Polling Technique

- CPU polls a certain I/O port continuously (using software) for data or readiness of the port to perform a data transaction.
  - For example, the CPU polls the printer port continuously to see if the printer is ready to accept data.
  - If it is ready, the CPU writes a data byte to the printer port. Else, it waits.
- CPU has full control and dedicate 100% of its resource in the whole data transfer process and does nothing else.



Polling Technique - Flowchart

- CPU performs all necessary initialization.
- CPU polls the I/O device for its readiness to perform data transfer.
- If I/O device is not ready, CPU continue to wait in the loop to check if device is ready.
- If device is ready, CPU make the data transfer and exit the loop.



Pros/Cons of Polling Technique

- Advantages
  - Programmer has complete control over the entire process.
  - Easiest method to test and debug.

done in software

- Disadvantages
  - Since the CPU waits in a loop, it cannot perform any other task until data transfer is completed.
  - Program execution of CPU held up while waiting for I/O device to get ready.
  - Inefficient use of CPU resources.

Interrupts

- A signalling mechanism that allows internal and external peripherals to alert the CPU that attention is needed.
- Could be in the form of a external/internal electrical pulse or change in internal register status.
- Once the CPU receive the signal, known as interrupt request, it would then need to decide if it wanted to service the request.
- If CPU decides to service interrupt request, it will follow up with the series of procedures to handle the interrupt event. ← both hardware & software
- Interrupt mechanism is typically used trigger the CPU to start some operation, e.g. data transfer to memory, status registers, control registers etc.

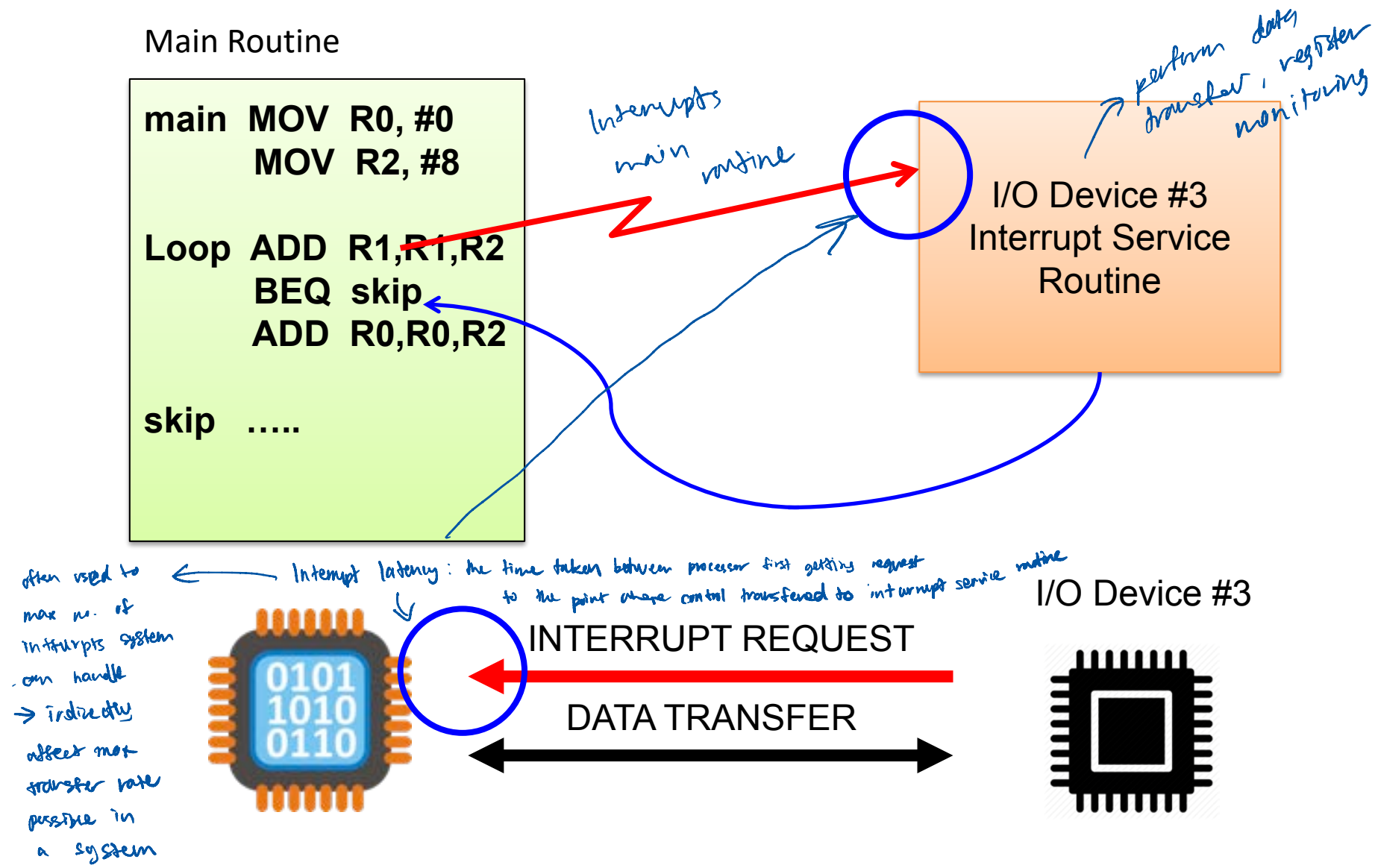
Polling vs Interrupt

- Boiling Water Analogy

Polling:
Check every few minutes

Interrupt:
Listen for the whistle

Interrupt Triggered Data Transfer



Interrupt Vector Table

Interrupt Vector Table

0	Reset ISR Addr
1	…
2	…
3	…
4	…
5	…
6	…
7	Ext INT 3 ISR Addr
8	SPI ISR Addr
9	UART ISR Addr
A	

- How does the CPU know the location of the corresponding interrupt service routine for each interrupts?
- The starting address of each interrupt is stored in a table known as the interrupt vector table
- Each interrupt has a unique index to the vector table, e.g. UART interrupt could be in index 9, so if a UART interrupt occurs, CPU will know where to branch to by checking index 9 in the vector table.
- The values and interrupt source in the table on the left are for illustration purpose only, different processor has different indexing for their interrupts.

Interrupt Control Flow

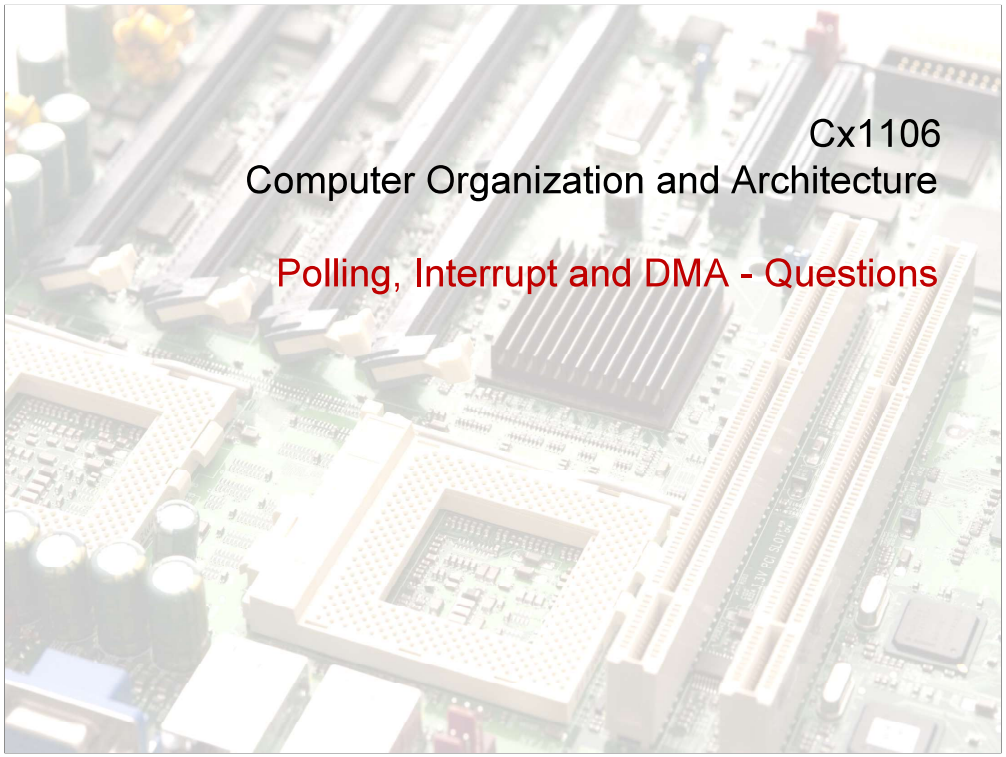
- A signal from external/internal peripheral, or a change in status of some special registers notifies the CPU that some event has occur and ask for CPU's (immediate) attention.
- If the CPU decide to service the interrupt, it will suspend its current program temporarily.
- CPU look up the interrupt vector table to check the starting address of the interrupt service routine (ISR) for the corresponding interrupt.
- CPU would save a copy of the processor context, i.e. the current value of various registers, this is to allow the interrupted routine to continue its execution after returning from the ISR.
- CPU then proceed to execute the ISR linked to the interrupt that was triggered.
- Once the ISR is completed, CPU restore the save context, returns to the interrupted routine and continue from where it had left previously.

Interrupt Service Routine

- The ISR will perform some operation e.g. Data Transfer.
- ISR is typically a very short routine so as not to suspend the main program for too long.
- After servicing the ISR, CPU will return to the previous program and continue from it branched off.
- It is possible for CPU to receive multiple interrupt requests simultaneously since it typically interface to multiple devices.
- In this case, some arbitration scheme has to be designed to decide which interrupt to service first (priority, first-come-first-serve etc).

Pros/Cons of Interrupt Triggered Technique

- Advantages
  - Efficient use of CPU resources as it does not need to monitor I/O device status.
  - CPU can continue with other tasks between interrupts.
  - Allows prioritization and pre-emption. → high priority events would take over lower priority events
    ↳ e.g. fire alarm more impt than normal temp taking
- Disadvantages
  - More hardware interface circuitry required between I/O device and processor. ↳ "black box" to program
  - Program is slightly more complex and difficult to debug.



Cx1106

Computer Organization and Architecture

Polling, Interrupt and DMA - Questions

Which of the following statement is FALSE?

A. Where responding to the intended trigger event is concerned, polling based mechanism would be able to respond faster than a interrupt based mechanism.

B. In Polling based mechanism, CPU has complete control over the entire trigger event detection process.

C. Interrupt based system is able to execute tasks more efficiently.

✓D. Its easier to debug interrupt based application since the initial process is handled by the hardware.

[A] Polling is able to yield faster response if the software is dedicated to the target status. Just that it'll take up 100% of the CPU.
[B] Polling is software based. User have full control.
[C] In Interrupt based system, CPU is able to proceed with its task and doesn't need to monitor the status constantly.
[D] When a process is handled in hardware, the status of the operation is not as readily available, typically only via some status registers. So when issue occur, there are less information available to the user and therefore more difficult to debug.

Which is the correct sequence of event for a interrupt control flow?

a: CPU suspend current program
b: Device send interrupt request
c: CPU return to interrupted program
d: CPU branch to Interrupt Service Routine
e: CPU decides to service the interrupt request

A. b, a, c, d, e

B. b, a, e, d, c

✓C. b, e, a, d, c

D. b, e, d, a, c

Why is interrupt based code more difficult to debug?

A. Interrupt mechanism involves more hardware.

B. More difficult to observe the state of hardware when issue occurs

C. Interrupts are asynchronous in nature, i.e. can occur anytime

✓D. All of the above

Which of the following information about DMA is FALSE?

A. DMAC is a separate module in a computer system used to relief CPU from the task of transferring data.

✓B. Transferring data via DMA is always faster than using CPU.

C. CPU response time will be the slowest when Burst mode DMA is used because the DMAC will transfer the entire block of data before releasing the system bus.

D. Transparent mode DMA results in the slowest transfer rate as it will only operate when the CPU is not using the system bus.

Transferring via CPU can be faster than DMA as CPU speed is typically faster, just that do so means wasting CPU resources to just transfer data.

**What is/are the advantages of using a DMAC instead of a CPU to transfer data?

A. DMAC is a separate bus controller

✓B. CPU does not need to perform the data transfer task and can focus on doing data processing.

✓C. DMAC is more efficient in performing data transfer for data with specific formatting, e.g. audio and video data.

D. DMAC is always able to transfer data at a faster rate compared to CPU.

What are the basic parameters that DMAC needs to perform a data transfer?

A. source address, destination address, read/write operation, end symbol

B. source address, number of bytes, destination address, end symbol

✓C. Source address, destination address, trigger signal, number of bytes

D. Source address, trigger signal, number of bytes, end symbol

If your application requires good system response with fairly good data transfer capability, which DMA mode would you use?

A. Transparent

✓B. Cycle Stealing

C. Burst

Which DMA transfer mode is capable of sustaining the fastest transfer rate?

A. Transparent
B. Cycle Stealing
✓C. Burst

**Which of the following determine the DMA data transfer rate of a system?

✓A. Transfer rate of the slowest module
✓B. Type of DMA transfer mode used
✓C. CPU's System Bus utilization rate
D. Transfer rate of the faster module

This question assume that DMAC and CPU share one system bus. So when if CPU uses the system bus frequently, that means DMAC will have less chance to use the bus and will have to transfer data at a slower rate.

In general, transfer rate among a group of modules will be limited by the slowest module in the group. Very much like the speed of a convey will be limited by the slowest vehicle in the convoy.