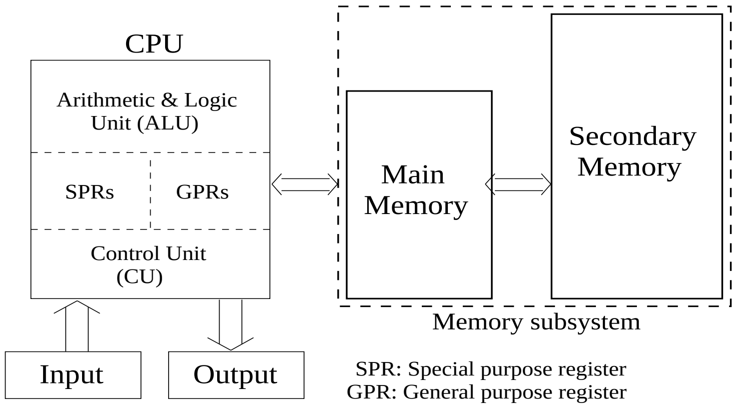
Lecture 23-24: April 9, 2021
Computer Architecture and Organization-I
Biplab K Sikdar

Peripherals


CPU
Arithmetic & Logic Unit (ALU)
SPRs
GPRs
Control Unit (CU)
Main Memory
Secondary Memory
Memory subsystem
Input
Output
SPR: Special purpose register
GPR: General purpose register

A machine computer without an input/output device is of no use.

Conventionally, CPU data processing speed is much much faster than input to CPU/CPU to output data transfer rate.

0.1 Properties

External I/O device normally communicates with I/O controllers interfaced with CPU through system bus (also used for data transfer between CPU and memory).

A system bus is shown in in Figure 1.

Command signals denote operations to be performed such as *memory read*, *memory write*, I/O *read*, I/O *write*, *transfer ACK*[1], *bus request*[2], *bus grant*[3], *interrupt request*[4], *interrupt ACK*[5], *clock*, *reset*, etc.

Performance of a bus structure is measured by defining its *bandwidth*.

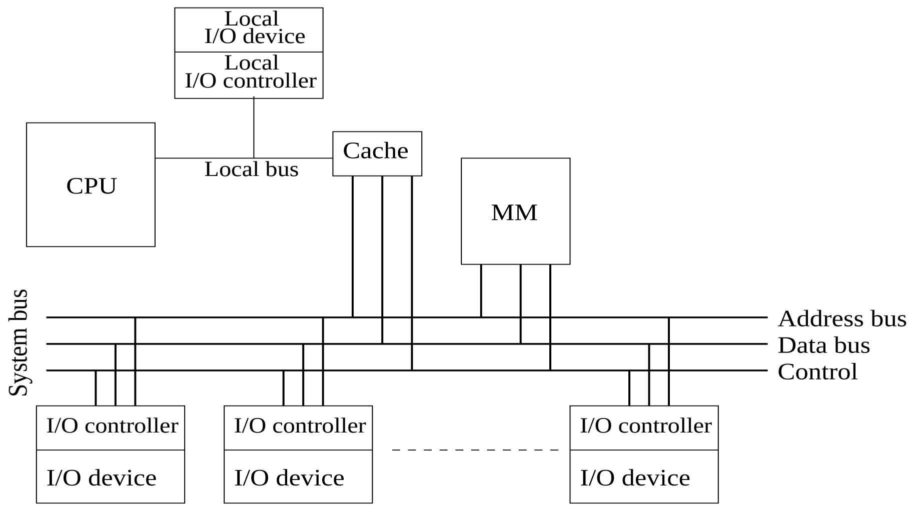


Figure 1: System bus

Such single-bus architecture provides uniform interfacing for devices (Figure 1).

It increases probability of bottleneck during data transfer.

[1]Indicates that data have been accepted from or placed on the bus.

[2]Indicates that a module needs to gain control of the bus.

[3]Indicates that a requesting module has been granted control of the bus.

[4]Indicates that an interrupt is pending.

[5]Acknowledges that the pending interrupt has been recognized.

0.1.1 Multiple-bus structure

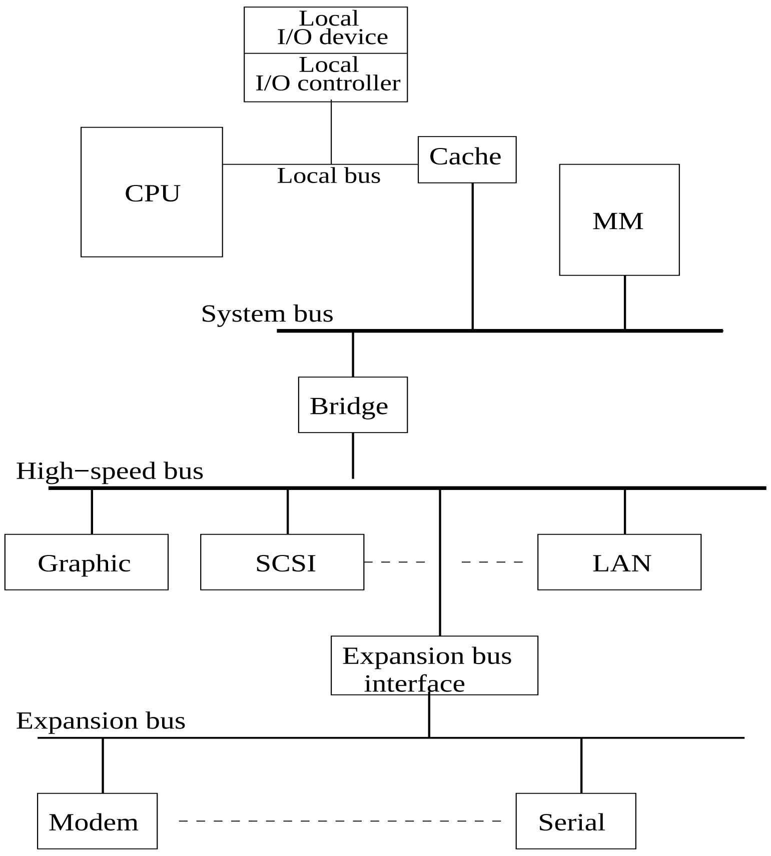


Figure 2: Multiple-bus structure

Multiple-bus structure (Figure 2) improves overall bandwidth.

It insulates memory-to-CPU traffic from I/O traffic.

SCSI (small computer system interface) supports disk drives and other peripherals.

The serial port is used to support printer/scanner.

Limitation : complexity of hardware and power consumption increases

Peripheral component interconnect (PCI), from Intel, is processor-independent bus.

PCI and PCI Express (PCIe) deliver better system performance for high-speed I/O.

0.1.2 Problems with the I/o interfacing

Reasons for which the peripherals are not directly connected to system bus are:

1. Wide variation of I/O devices - of type mechanical, electrical, electromechanical and electronic. Even transducers, ADCs or DACs are used.
2. Speed variations
3. Wide variation in format of data : serial, parallel, ...

0.1.3 Data transfer schemes

Data transfer between micro-processor and I/O, and memory and I/O devices.

Timing incompatibility among processor and I-O devices may cause problems.

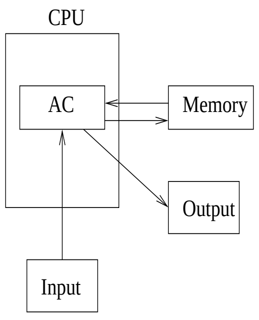


Figure 3: Programmed I/O

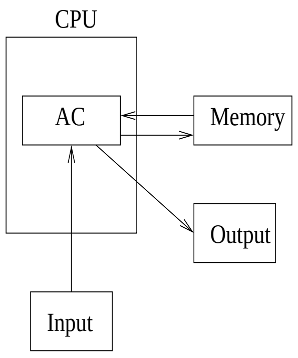
i) Programmed I/O
ii) DMA (direct memory access).

DMA bypasses CPU.

0.2 Programmed I/O Data Transfer

CPU executes program - it results in data transfer (between I/O device and memory).

Completely controlled by CPU and it is (i) from input to CPU (say, AC) and from CPU to memory; or (ii) from memory to CPU and then from CPU to output device.



Three alternative schemes in programmed I/O -

1. Synchronous (mode) data transfer
2. Asynchronous (mode) data transfer
3. Interrupt driven (mode) data transfer

0.2.1 Synchronous mode of data transfer

CPU is aware of speed of I/O device.

CPU synchronizes its own speed with that of I/O device speed.

CPU and I/O device share a common clock.

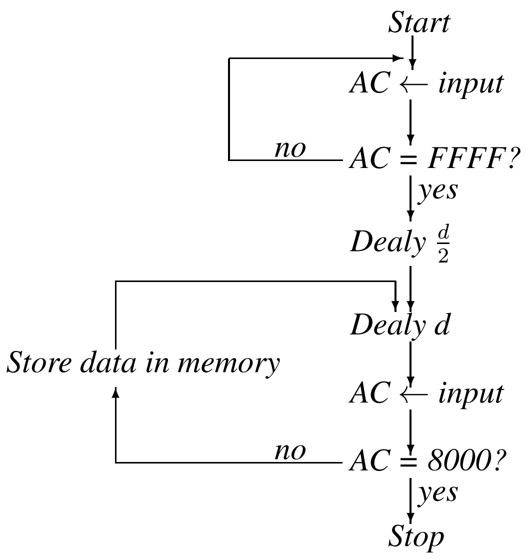
Example 0.1 A device transmits 16-bit data packet with start of transmission as all "1"s code and 8000_H at end of transmission (packet).

Device prepares a data at a delay of d time unit.

Synchronous data transfer scheme is shown in Algorithm 0.1.

Delay of $\frac{d}{2}ms$ in Algorithm 0.1 is to get steady state data.

Algorithm 0.1 *Synchronous data transfer*



May be used when I/O device and a processor match in speed.

0.2.2 Asynchronous/Handshaking mode of data transfer

Is effective when processor/CPU does not know the speed of I/O device.

Suitable for low-speed I/O devices.

An unit can use its own private clock.

CPU checks whether input [output] device is ready to send [accept] next data.

Follow Figure 4.

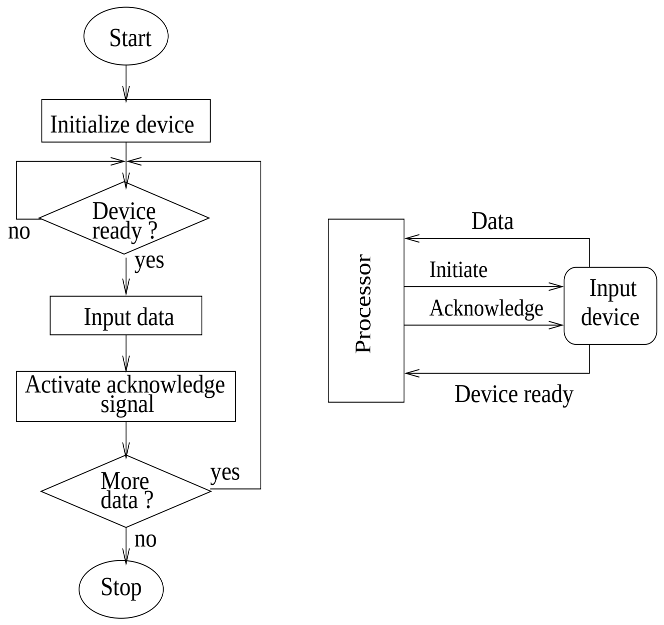


Figure 4: Asynchronous data transfer

Step 1. When input device is ready to transfer new data it outputs the data on data bus and sends a data ready signal

Step 2. CPU continuously tests status of data ready signal until it is active. When active CPU reads in data from data bus, and sends acknowledgement signal

Step 3. When input device receives acknowledgement signal, it deactivates data ready line, and goes bock to Step 1

Example 0.2 Data transfer from analog to digital converter to processor (Figure 5).
SOC: start of conversion, EOC: end of conversion of analog input (V_A).
a pulse is computed in this line when conversion is complete.
CPU always checks EOC to read data.

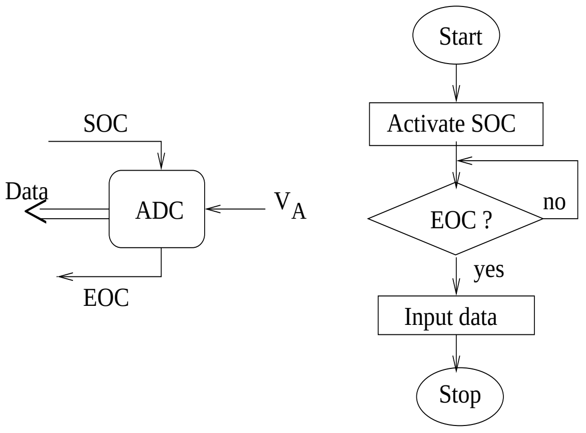


Figure 5: Asynchronous data transfer from ADC to CPU

CPU remains busy to get next data -that is, CPU waste time and can’t do other job.

0.2.3 Interrupt driven data transfer

CPU can proceed to execute its job/program.

For data transfer from input device to CPU, input device issues an interrupt request.

When CPU receives device ready signal through an interrupt it process I/O transfer routine and then returns to the task it was originally performing.

Summary of the steps to be executed is given below.

Step 1. CPU initializes I/O devices (A/D converter).

Step 2. CPU continues excution of subsequent instructioni in the program.

Step 3. I/O device sends signal (interrupt) to CPU when it is ready.

Step 4. CPU completes execution of current instruction and grants the interrupt.

(a) It saves PC on a stack.

(b) Branches to location IBA (Int. Branch Add) where ILS, ISS are stored.

(c) CPU executes ISS, saves processor registers, stack pointer etc on stack.

(d) CPU then transfers data.

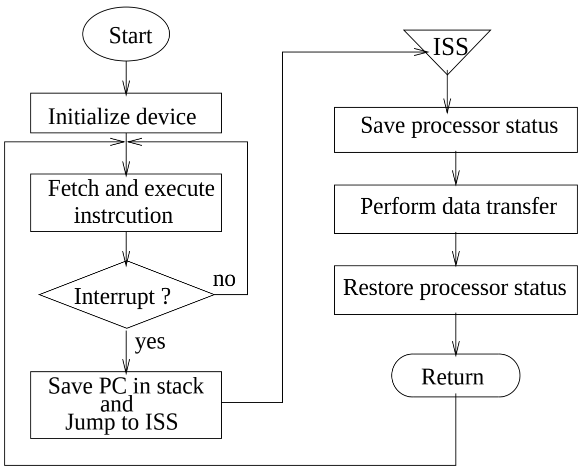


Figure 6: Interrupt driven data transfer

This scheme is effective for relatively slow I/O devices (keyboard/ADC/DAC/printer/etc).

0.3 DMA

An external device - DMA controller (DMAC), on behalf of CPU, transfers data directly from input device to memory as well as memory to output device.

DMA transfer takes place without execution of IO instructions by CPU.

Prior to a data transfer, CPU has to initialize DMA controller with:

1. Length of the block to be transferred,
2. Starting address of memory (to) from which data transfer should occur.

During the DMA data transfer the CPU is forced to ”hold on”.

CPU suspends its current activities and attends DMA request.

CPU can respond to a DMA request after current m/cycle (Figure 8).

CPU releases system bus and signals by activating DMA acknowledgement signal.

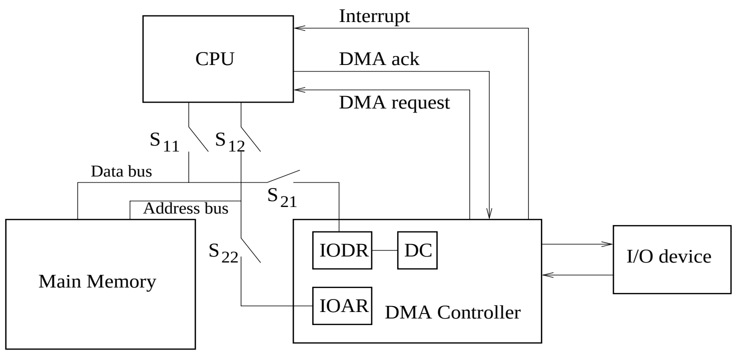


Figure 7: DMA interfacing

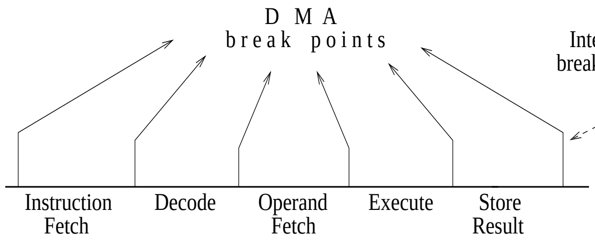


Figure 8: DMA break points