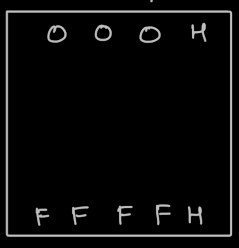
I/O device → memory device
I/O device

memory mapped I/O

I/O mapped I/O.

memory address space



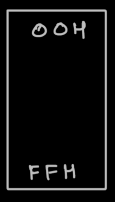
16 bit address

2^{16} = 64 KB

(65,536 address)

Ram Rom get this address.

I/O address space.



8-bit address → because there are less I/O devices.

$\therefore 2^8 = 256$ address. → still overkill.

mapped under I/O space. → This technique is I/O mapped I/O.

Def^n: I/O device mapped into I/O space.

I/O device gets I/O address.

Processor differentiates memory and I/O

8 bit I/O address.

256 possible I/O devices.

Only instructions that can be used are IN & OUT.

Only A reg. can be used to transfer data.

Since there are 2 different entities memory & I/O

4 different instructions are required

MR MW IO R IOW.

↓

Smaller add = easier to decode add.

So simple, quicker, cheaper add. decoding.

eg: microprocessor like 8085, 8086 by default work on this method.

I/O device mapped into memory space

I/O device given memory add.

Donot differentiate memory & I/O.

16-bit add. given.

2^{16} = 63,536 ★★

Bunch of instructions work now on I/O.

Can be connected to any Reg.

Only 2 signals required. Read & write

eg: 8051 microcontroller.

Complex, slower, Expensive to decode addresses.

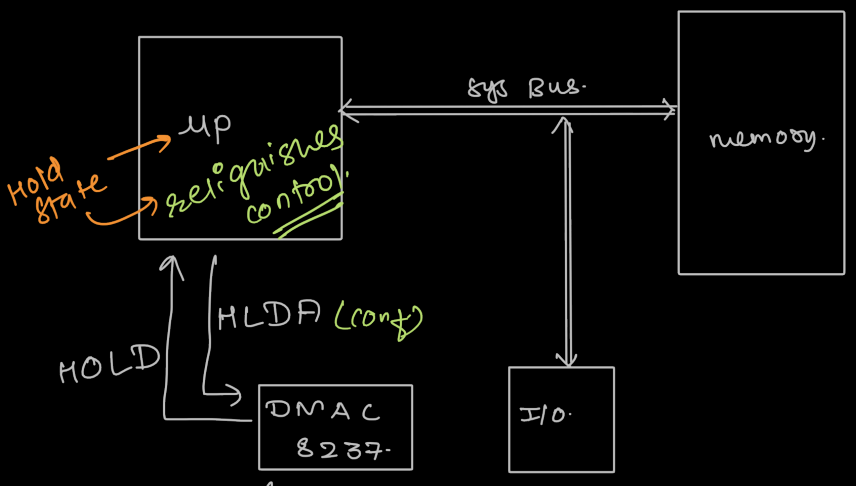
↓

Industrial purpose

eg: Lightshow in Burj Kalifa.

DMA: Direct memory access.

- transfer of data between memory and I/O.
- advantage: superfast transfer speed.



generates control signal donot interfere with the data so can be a bus master.

- Two device can transfer only if either of them can generate control signal.
- so up required.

I/O ⟶ memory.

- I/O Read
- memory write.

} 2 instruction required.

disadvantages seen while transferring large files.

To transfer data by normal way:

µp has to fetch instruction, decode it and execute it
transfer happens in execution.
There are 5 instructions in transfer 1 byte.

- read data
- send data
- add. incr.
- dec. counter
- loop

} DMA solve the time waste during fetching & decoding.

When does DMAC send HOLD signal?.

processor informs the DMA controller.
it provides CAR & CWCR

DMA controller
- CAR - Current Add. Reg. (Name)
- CWCR - Current word count Reg. (Size)

⟶ DREQ - Data Request
↳ sent by I/O device to confirm it is ready.
by default it is 1.

if 1 sends HOLD signal.
if 0 (service routine)

just before transfer DMAC sends $\overline{DACK}$ (data ack.) to I/O device to wake them up (power saving).

Transfer - 1 byte 1 cycle. until terminal count.

increamenter of add.
decreamenter of count

} is default. so no fetching decoding required. ⟶ hardware based.

DMA Transfer Mode

a) Burst Mode / Block Xsfer

{Dead terminals like servers}

- after DMA becomes bus master, it transfer whole block in 1 brust
- only returns control after whole transfer is complete.

adv. — fastest form of DMA.

disadv. up is in hold state for long period of time.

b) Cycle Stealing Mode / 1 byte Xsfer.

- after DMA becomes bus master, transfer 1 byte returns control to up, up performs cycle and again DMA performs another cycle hence cycle stealing.
- dis. — method is slow

adv. — up is still active so multi-tasking is possible

modern day up use this method.

c) Demand Xsfer

It is an added feature.

DREQ has to be 1 throughout the transfer meaning there has to be constant demand.

eg: Printer → it has to stop the flow of data.

d) Transparent Mode / Hidden Mode

— After DREQ = 1 DMA waits until up is idle to send HOLD signal.

$\overline{RD} = 1$ $\overline{WR} = 1$ {inactive}

— while transfer is going on, any moment on up DMA leaves the control to up so transparent to up. and processes of up is not disturbed.

dis adv. — hardware req. can be expensive

- slower transfer.

8237 DMA Controller & Interfacing.