Lecture 11 (I/O devices)

I/O devices:

- block (fixed-size block with own address, transfers units = entire block) - hard disks
- character (stream of characters, doesn't addressable, no seek operation) - printers, mice

Out of model devices: clocks, memory-mapped screen, touch screens

I/O = mechanical component(device itself) + electronic component(device controller). Interface between controller & device - low-level

Preamble written to disk when its formatted contain cylinder, sector number & synchronization info

Controller convert serial bit stream to block of bytes & check errors. Block assembled bit by bit, checksum verified(block error free) & it copied to main memory

---

Memory-mapped I/O

Controller has registers for communicating with CPU & may have data buffer. OS writes to register = command device, OS reads from registers = know device state.

Two ways of communicating between CPU & device (control register)

- Control register assigned to I/O port number. All I/O ports form I/O port space. Users cannot access port space, only OS. (Memory & I/O ports in diff places)
- **Memory-mapped I/O** - each control register is assigned a unique memory address to which no memory is assigned (mapping between memory space & control registers). Assigned addresses at top of address space or near.
- Hybrid scheme (memory-mapped data buffer & separate I/O ports for control registers.

Algorithm: CPU puts address on bus' address line→ asserts READ on bus' control line → second signal line determine I/O space or memory space → memory space(memory responds to request) or I/O space(I/O device responds to request).

Advantages of memory-mapped I/O:

- written in C
- no special protection mechanism is needed to keep user processes from performing I/O
- OS can give user control to some specific devices by including its pages into page table
- every instruction can reference memory & control registers simultaneously.

Disadvantages of memory-mapped I/O:

- caching of memory words(taking value from cache without asking device) - solved by selectively disabling caching
- If one address space ⇒ examine all memory reference to determine which belongs to memory & which belongs to I/O

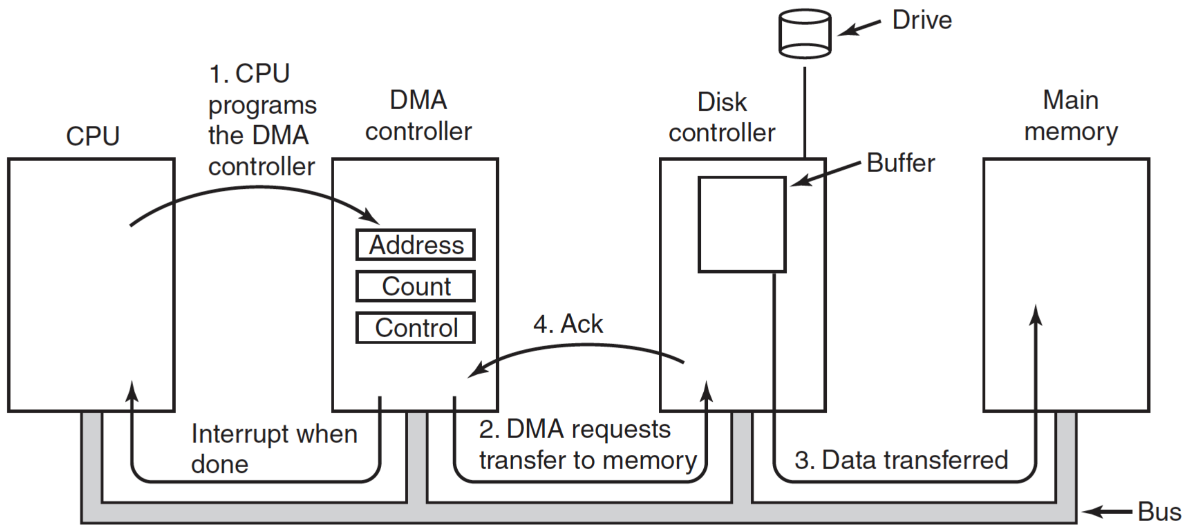
DMA(Directed Memory Access)

DMA allows hardware access main memory independent of CPU.

DMAC has access to system bus independent on CPU. DMAC registers: memory address register, byte count register, control registers(I/O port, direction of transfer, transfer unit, number of bytes for transfer in one burst).

Disk Read Occur w/o DMA: disk controller reads block bit by bit to internal buffer → compute checksum(error checking) → OS reads block by units from controller's buffer

Disk Read Occur with DMA: CPU programs DMAC registers → checking errors → DMAC transfer by read request over bus to disk controller → write to memory in standard bus cycle → disk controller to DMAC acknowledgment signal if complete → DMAC interrupt to CPU



Bus modes:

- Word-at-time mode (DMAC request transfer of one word & get it; if CPU wants bus ⇒ CPU waits = **cycle stealing**) - small data with regular frequency (interrupt from keyboard)
- Block mode (DMAC tells device acquire bus, issues series of transfers, release bus = **burst mode**, burst mode more efficient than cycle stealing, burst mode can block CPU & devices for substantial period if long burst) - lot of memory read from harddisk
- Fly-by mode (DMAC tells device to transfer data directly to main memory)

Transfer using physical memory: convert virtual address of memory buffer to physical address → write physical address to DMAC's address register. Alternative: virtual address to DMAC's address register → DMAC use MMU to translate address to physical address.