Input-Output Interface and Memory Organization

Computer Architecture — Module 5 Notes

1. Input-Output Interface

1.1 Introduction to I/O Interface

The I/O interface provides a communication pathway between the CPU and external devices (peripherals).

Why I/O Interface is Needed:

- Speed Mismatch: CPU is much faster than I/O devices
- Data Format: Different data formats (serial vs parallel)
- Voltage Levels: Different voltage requirements
- Data Transfer: Different transfer rates
- Protocol: Different communication protocols



1.2 Types of I/O Interface

1. Programmed I/O:

- CPU continuously checks I/O status
- Polling method
- Simple but inefficient

2. Interrupt-Driven I/O:

- Device interrupts CPU when ready

- More efficient than polling
- Requires interrupt handling

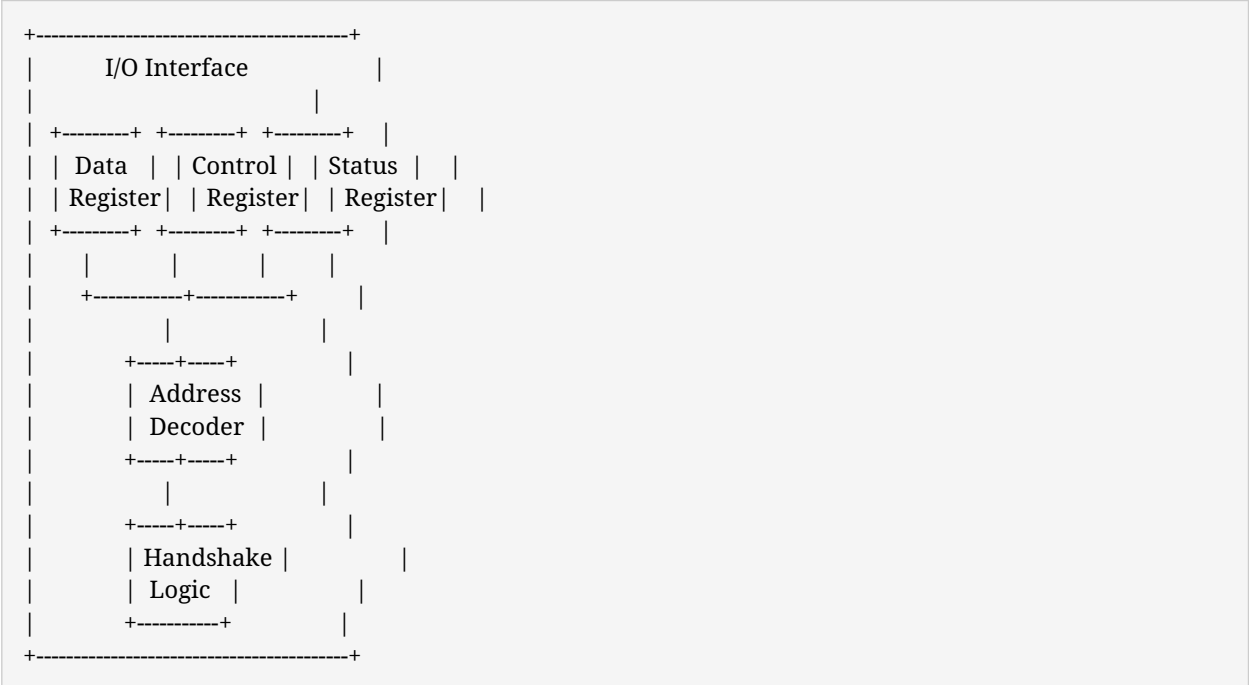
3. Direct Memory Access (DMA):

- Data transferred directly between I/O and memory
- CPU involvement minimal
- Most efficient for large data transfers

1.3 I/O Interface Components

Component	Description	Function
Data Register	Holds data for transfer	Buffer for data
Control Register	Controls I/O operations	Device configuration
Status Register	Indicates device status	Ready, busy, error
Address Decoder	Decodes I/O addresses	Selects device
Handshake Logic	Coordinates transfer	Synchronization

Example I/O Interface Structure:



2. Asynchronous Data Transfer

2.1 Synchronous vs Asynchronous Transfer

Aspect	Synchronous	Asynchronous
Clock	Common clock	No common clock
Timing	Based on clock edges	Based on handshake
Speed	Faster	Slower
Complexity	Simple	More complex
Data Rate	Fixed	Variable

2.2 Handshaking Protocol

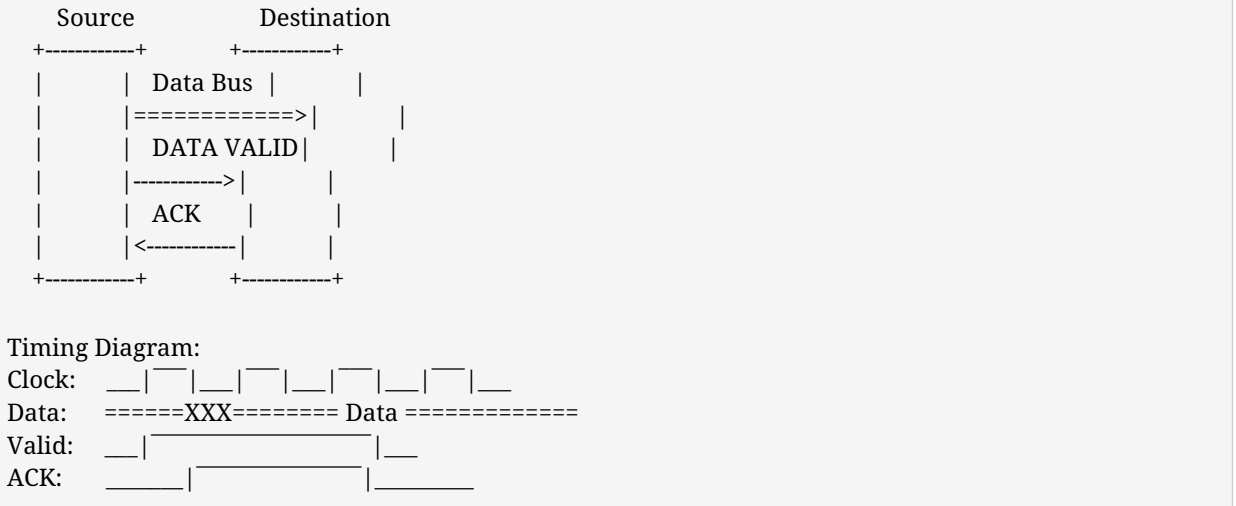
Handshaking: A method of coordinating data transfer between devices.

Strobe Control:

- A single control line indicates when data is valid
- Simple but less reliable

Full Handshake:

- Uses two control signals (DATA VALID and ACKNOWLEDGE)
- More reliable



2.3 Handshaking Steps

Source-Initiated Transfer:

1. Source places data on bus

```
2. Source asserts DATA VALID signal
3. Destination reads data
4. Destination asserts ACKNOWLEDGE
5. Source deasserts DATA VALID
6. Destination deasserts ACKNOWLEDGE
```

Destination-Initiated Transfer:

```
1. Destination requests data
2. Source places data on bus
3. Source asserts DATA VALID
4. Destination reads data
5. Destination deasserts request
6. Source deasserts DATA VALID
```

2.4 UART (Universal Asynchronous Receiver/Transmitter)

UART Frame Format:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Start| Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Stop | Stop |
| 0   | LSB |     |     |     |     |     | MSB | 1   | 1   |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

Components:
1. Start bit (0)
2. Data bits (5-8 bits)
3. Parity bit (optional)
4. Stop bits (1 or 2)
```

3. Modes of Data Transfer

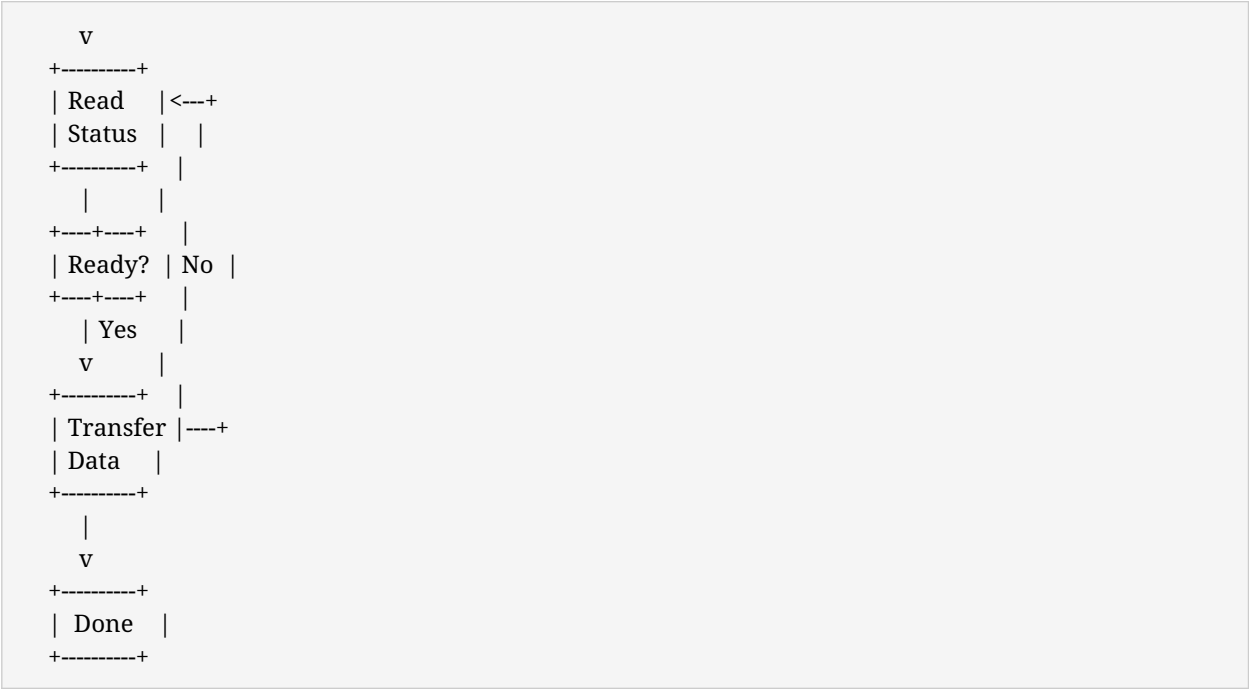
3.1 Programmed I/O (Polling)

Characteristics:

- CPU actively checks I/O status
- Simple implementation
- Wastes CPU time
- Suitable for slow devices

Flowchart:

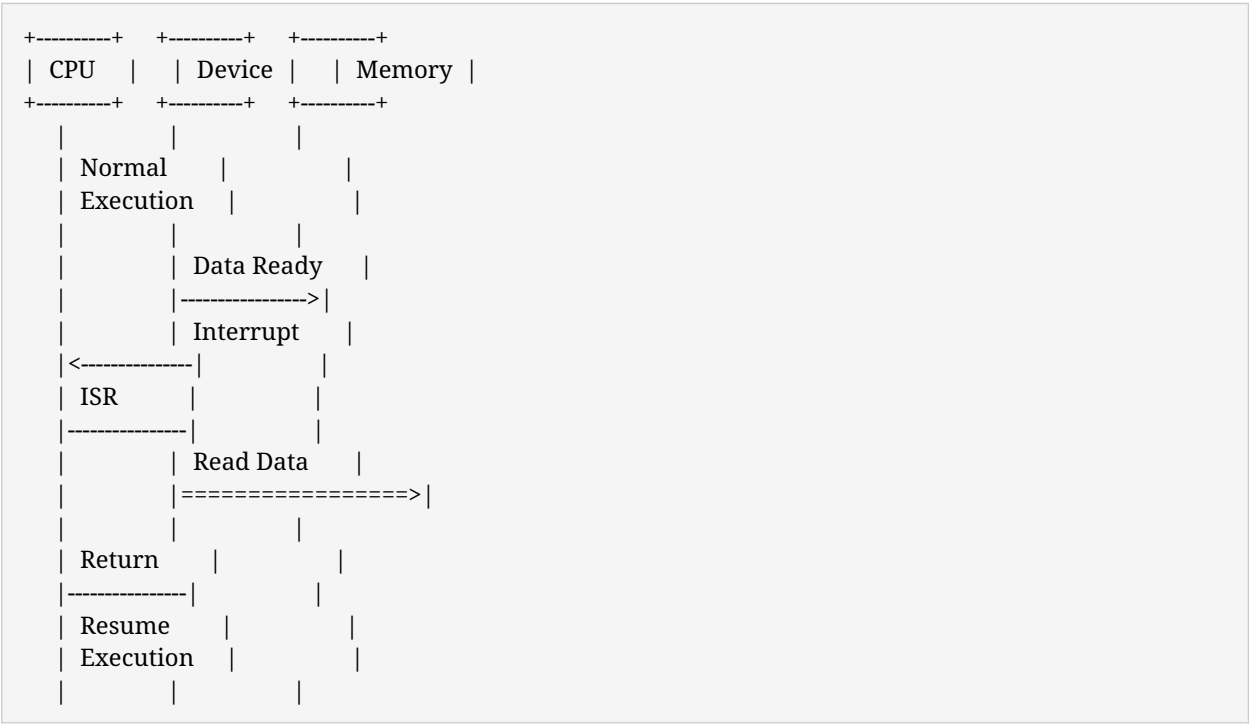
```
  +----------+
  | Start   |
  +----------+
     |
```



3.2 Interrupt-Driven I/O

Characteristics:

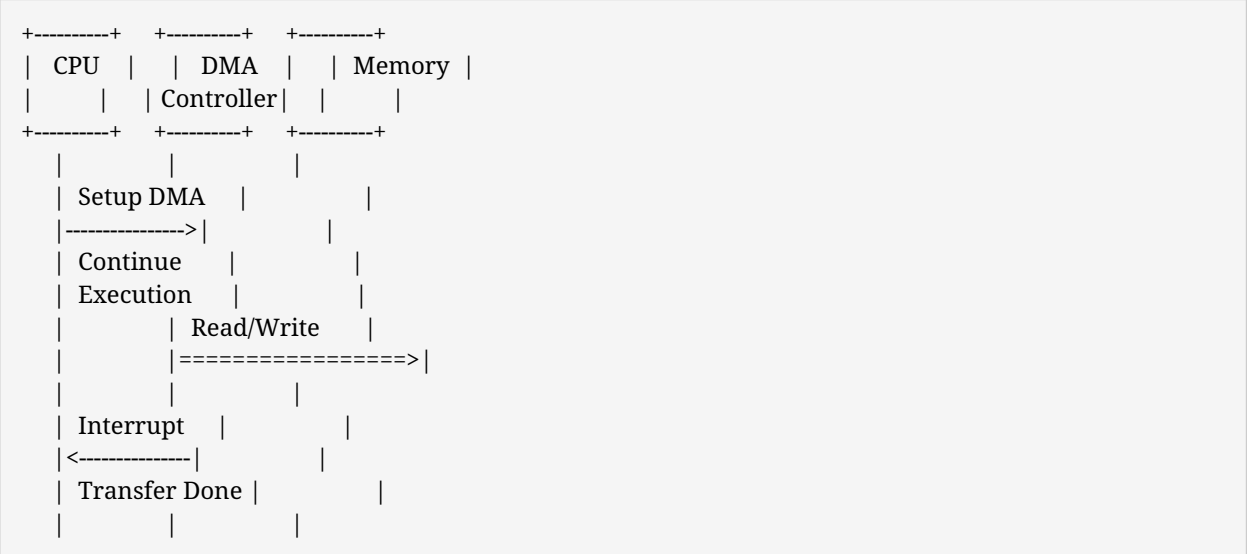
- Device interrupts CPU when ready
- CPU can do other work
- More efficient than polling
- Requires interrupt controller



3.3 Direct Memory Access (DMA)

Characteristics:

- Data transferred directly between I/O and memory
- CPU sets up transfer, then continues
- DMA controller manages transfer
- Most efficient for large transfers



3.4 Comparison of I/O Modes

Feature	Programmed I/O	Interrupt I/O	DMA
CPU Involvement	High	Medium	Low
Efficiency	Low	Medium	High
Complexity	Simple	Medium	Complex
Suitable For	Simple devices	Medium speed	High speed
CPU Time	Wasted	Efficient	Minimal
Data Size	Small	Medium	Large
Examples	Keyboard, LED	Mouse, Keyboard	Disk, Network

4. Priority Interrupt

4.1 Interrupt Priority System

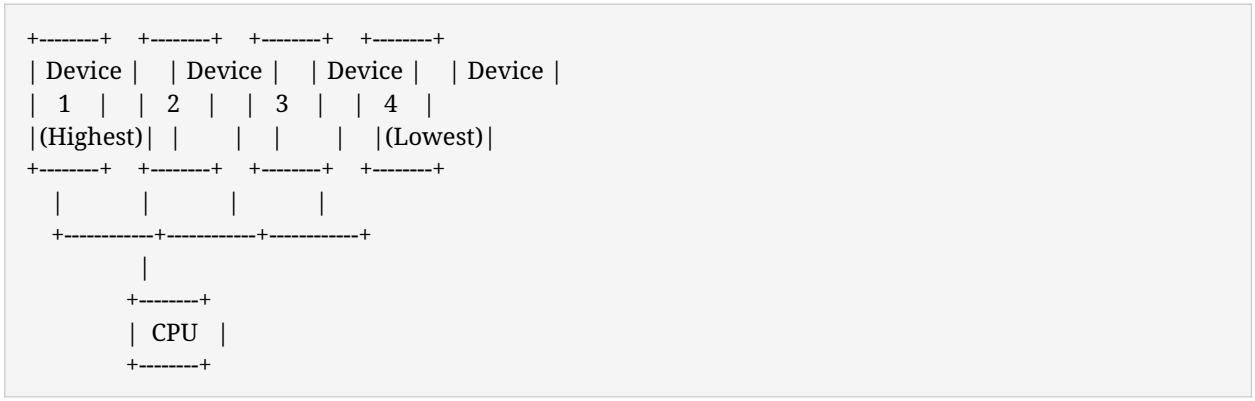
Why Priority Interrupt?

- Multiple devices may interrupt simultaneously
- Some devices need faster response
- Critical tasks must be handled first

Priority Schemes:

- Fixed Priority: Predefined order
- Daisy Chain: Hardware priority chain
- Polling: Software priority check

4.2 Daisy Chain Priority



Working:

1. Interrupt request travels through chain
2. Highest priority device accepts interrupt
3. Lower priority devices see busy signal
4. CPU processes highest priority device

4.3 Priority Encoder

Priority Encoder Table (8-to-3):

I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0	Valid
1	X	X	X	X	X	X	X	1	1	1	1
0	1	X	X	X	X	X	X	1	1	0	1
0	0	1	X	X	X	X	X	1	0	1	1
0	0	0	1	X	X	X	X	1	0	0	1
0	0	0	0	1	X	X	X	0	1	1	1
0	0	0	0	0	1	X	X	0	1	0	1
0	0	0	0	0	0	1	X	0	0	1	1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	X	X	X	0

4.4 Interrupt Vector

Interrupt Vector Table:

```
+--------+-------------------------+
| Vector | Interrupt Service Routine  |
| Number | Address                 |
+--------+-------------------------+
| 0      | Address of ISR 0        |
| 1      | Address of ISR 1        |
| 2      | Address of ISR 2        |
| ...    | ...                     |
| 255    | Address of ISR 255      |
+--------+-------------------------+
```

Interrupt Processing:

```
1. Save current context (PC, flags)
2. Determine interrupt source
3. Get vector address
4. Jump to ISR
5. Execute ISR
6. Restore context
7. Return to interrupted program
```

5. DMA (Direct Memory Access)

5.1 DMA Architecture

```
+----------+    +----------+    +----------+
|   CPU    |    |   DMA    |    |  Memory  |
|          |    | Controller|    |          |
+----------+    +----------+    +----------+
    |               |               |
    |   Bus         |               |
    |   Request     |               |
    |<--------------|               |
    |   Bus         |               |
    |   Grant       |               |
    |-------------->|               |
    |   Bus         |               |
    |   Hold        |               |
    |<--------------|               |
    |   Data Bus    |               |
    |===============|===============|
```

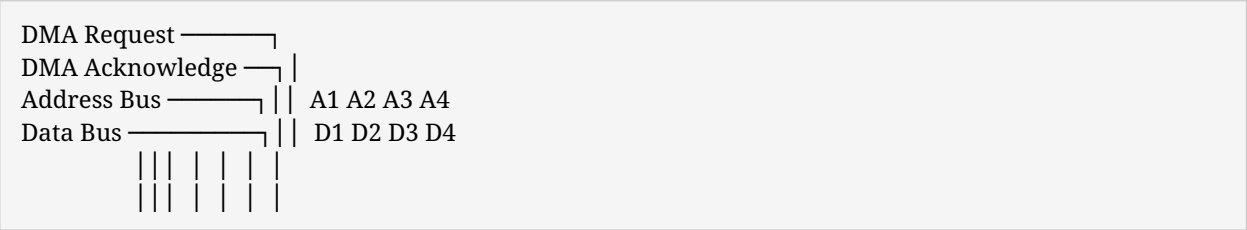
5.2 DMA Controller Components

Component	Function
Address Register	Holds memory address
Word Count Register	Number of bytes to transfer
Control Register	Transfer mode, direction
Status Register	Transfer status, errors
Data Register	Temporary data storage

5.3 DMA Transfer Modes

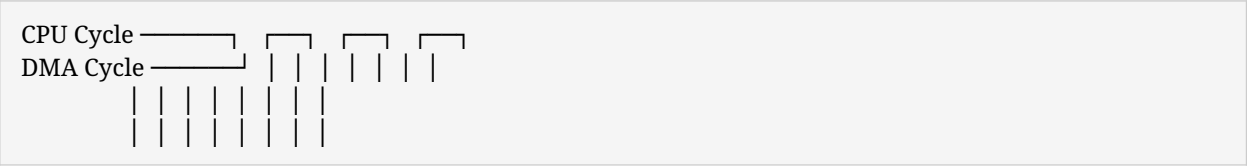
1. Burst Mode:

- Transfers entire block at once
- CPU is busy waiting
- Fastest transfer



2. Cycle Stealing Mode:

- One byte transferred at a time
- CPU gets memory access between transfers
- Slower but CPU remains active



3. Transparent Mode:

- DMA transfers when CPU is not using bus
- No interference with CPU
- Slowest transfer

5.4 DMA Configuration

Setup Process:

```
1. Load Memory Address Register
2. Load Word Count Register
3. Configure Control Register (direction, mode)
4. Enable DMA
5. Device requests DMA
6. DMA controller takes over bus
7. Transfer data
8. DMA controller signals completion
```

DMA Example:

```
+-------------------------------------------------+
| DMA Controller Configuration                    |
+-------------------------------------------------+
| Source Address:  0x1000                         |
| Destination:     I/O Device                     |
| Word Count:      1024 bytes                     |
| Mode:            Burst Mode                     |
| Direction:       Memory to I/O                  |
+-------------------------------------------------+
```

6. I/O Processor (IOP)

6.1 IOP Architecture

I/O Processor: A dedicated processor for handling I/O operations.

```
+----------+    +----------+    +----------+
|  CPU     |    |  IOP     |    | Memory   |
|          |    |          |    |          |
+----------+    +----------+    +----------+
     |               |               |
     |  Command      |               |
     |  Buffer       |               |
     |===============|===============|
     |               |               |
     |  Status       |               |
     |  Buffer       |               |
     |===============|===============|
```

6.2 CPU-IOP Communication

Communication Methods:

1. Memory-Mapped I/O:

- I/O registers mapped to memory addresses
- CPU uses memory instructions
- Simple implementation

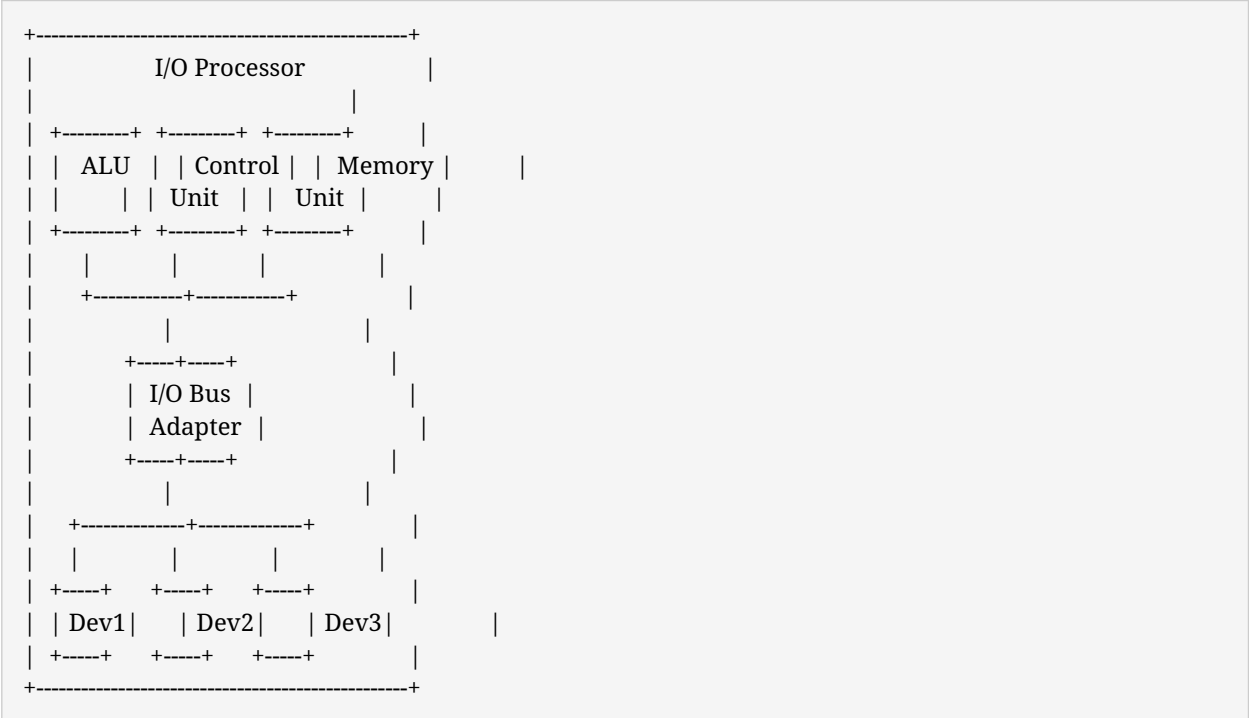
2. I/O-Mapped I/O:

- Separate I/O address space
- Special I/O instructions
- More complex but faster

6.3 IOP Functions

Function	Description
Command Processing	Executes I/O commands from CPU
Data Transfer	Handles data movement
Error Handling	Detects and handles errors
Buffer Management	Manages I/O buffers
Device Control	Controls I/O devices

6.4 IOP Architecture



7. Serial Communication

7.1 Serial vs Parallel Communication

Aspect	Serial	Parallel
Data Lines	1	Multiple
Speed	Lower	Higher
Cost	Lower	Higher
Distance	Longer	Shorter
Complexity	Simpler	More complex

7.2 Serial Communication Standards

RS-232:

- Voltage levels: ±12V
- Max distance: 15m
- Max speed: 20Kbps
- Uses: Modems, terminals

USB (Universal Serial Bus):

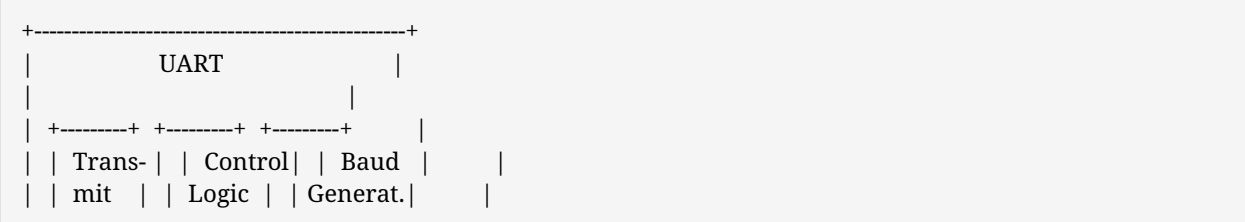
- USB 1.0: 1.5 Mbps
- USB 2.0: 480 Mbps
- USB 3.0: 5 Gbps
- Uses: Peripherals

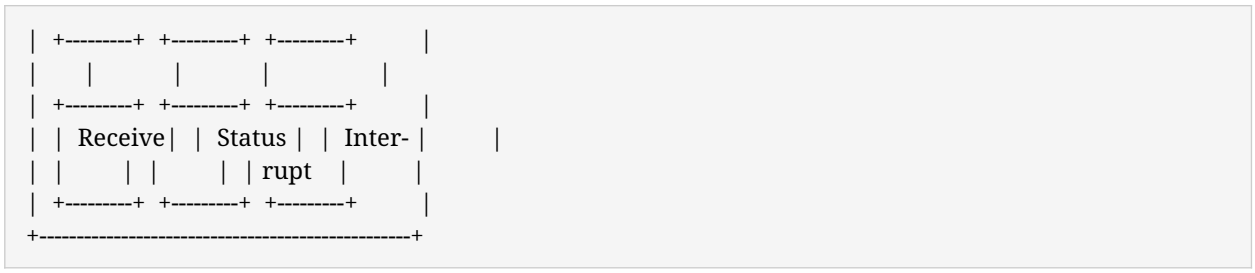
RS-485:

- Differential signaling
- Max distance: 1200m
- Max speed: 10 Mbps
- Uses: Industrial control

7.3 UART Working

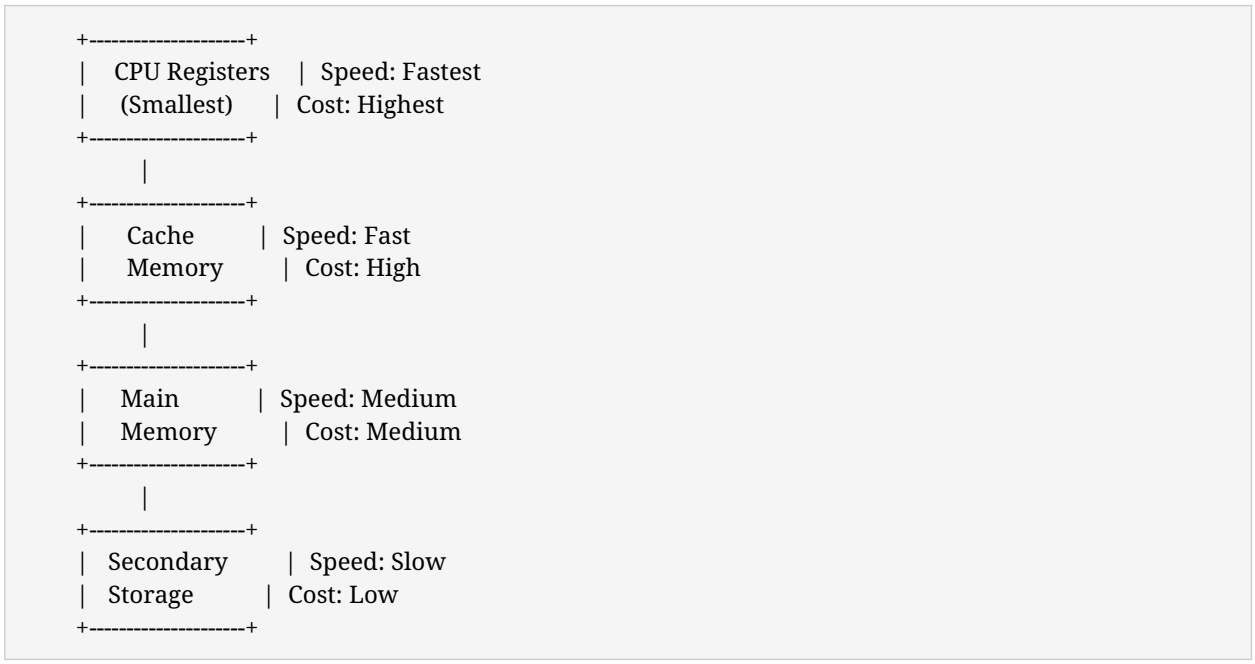
UART Block Diagram:





8. Memory Organization

8.1 Memory Hierarchy



8.2 Main Memory

Types:

1. RAM (Random Access Memory)

- SRAM (Static): Fast, expensive, cache
- DRAM (Dynamic): Slower, cheap, main memory

2. ROM (Read Only Memory)

- MROM: Mask programmed
- PROM: Programmable once
- EPROM: Erasable (UV)
- EEPROM: Electrically erasable

Memory Specifications:

```
Capacity: 4 GB, 8 GB, 16 GB
Speed: DDR3: 800-1600 MHz
       DDR4: 2133-3200 MHz
       DDR5: 4800-6400 MHz
Voltage: DDR3: 1.5V
         DDR4: 1.2V
         DDR5: 1.1V
```

8.3 Auxiliary Memory

Types:

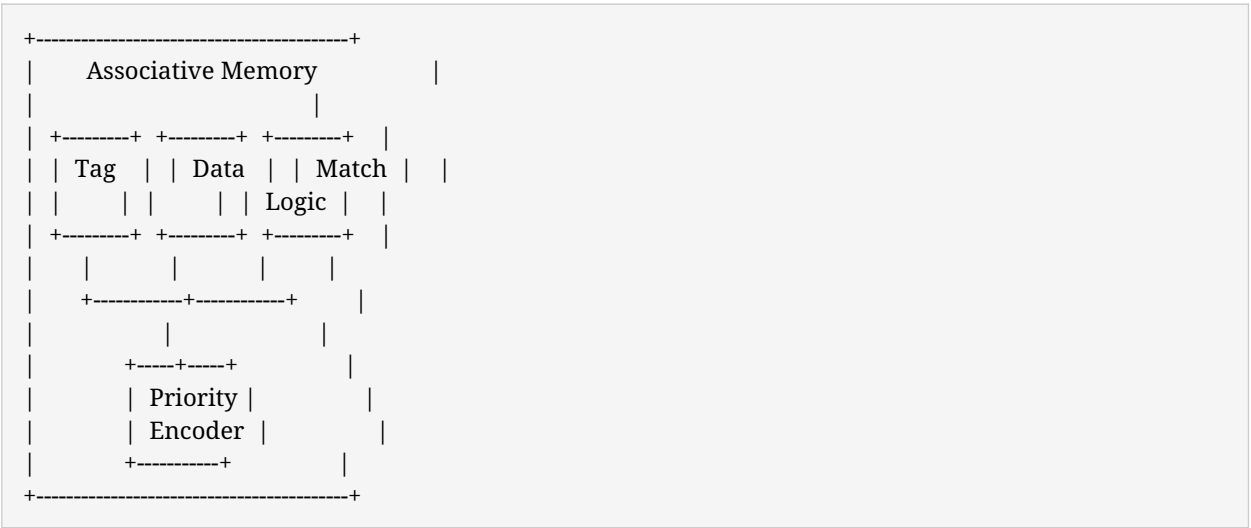
- Magnetic Disk: Hard disk drives (HDD)
- Solid State Drive: SSD (NAND flash)
- Optical Disc: CD, DVD, Blu-ray
- Magnetic Tape: Backup storage

Comparison:

Type	Speed	Cost	Capacity	Durability
HDD	Slow	Low	High	Low
SSD	Fast	Medium	Medium	High
Optical	Slow	Low	Medium	High
Tape	Very Slow	Very Low	Very High	High

8.4 Associative Memory

Concept: Content-addressable memory (CAM)



Operations:

- Search: Find data by content
- Read: Read data at matched address
- Write: Write data to matching location

8.5 Cache Memory

Cache Principles:

- Small, fast memory
- Stores frequently accessed data
- Reduces average memory access time

Cache Levels:

```
L1 Cache: 32-64KB (CPU core)
L2 Cache: 256-512KB (CPU chip)
L3 Cache: 4-8MB (CPU package)
```

Cache Mapping Methods:

- Direct Mapping: Fixed mapping
- Associative Mapping: Any block in any location
- Set-Associative Mapping: Blocks grouped into sets

Cache Replacement Policies:

- LRU (Least Recently Used)
- FIFO (First In First Out)
- Random
- LFU (Least Frequently Used)

8.6 Virtual Memory

Concept: Simulate more memory than physically available

Virtual Address Space:

```
+--------------------+
| Stack              |
| (Grows downward)   |
+--------------------+
| Heap               |
| (Grows upward)     |
+--------------------+
| Data Segment       |
```

```
+--------------------+
| Text Segment       |
| (Program code)     |
+--------------------+
```

Paging:

- Memory divided into pages (4KB typically)
- Page table maps virtual to physical pages
- Page fault when page not in memory

Page Table:

```
+--------+----------+----------+
| Page # | Frame #  | Present  |
+--------+----------+----------+
| 0      | 12       | 1        |
| 1      | 5        | 1        |
| 2      | -        | 0        |
| 3      | 8        | 1        |
| ...    | ...      | ...      |
+--------+----------+----------+
```

9. Memory Management

9.1 Memory Allocation Strategies

Strategy	Description	Advantages	Disadvantages
Fixed Partition	Memory divided into fixed blocks	Simple	Wasted space
Dynamic Partition	Variable block sizes	Efficient	External fragmentation
Paging	Fixed-size pages	No external fragmentation	Page table overhead
Segmentation	Variable-size segments	Logical structure	External fragmentation

9.2 Memory Protection

Protection Mechanisms:

- Base and Limit Registers
- Memory Protection Keys
- Memory Management Unit (MMU)

Base and Limit Registers:

```
+-----------------+
| Base Register   | Points to start of segment
|  (Base Address) |
+-----------------+
| Limit Register  | Size of segment
|  (Limit Value)  |
+-----------------+
```

Address Calculation:

```
Physical Address = Base Address + Virtual Address
if (Virtual Address >= Limit Register): Fault
```

9.3 Cache Memory Optimization

Cache Coherence Protocols:

1. MESI Protocol:

State	Description
Modified (M)	Cache has latest data, exclusive
Exclusive (E)	Cache has data, not modified
Shared (S)	Shared with other caches
Invalid (I)	Cache line is invalid

2. Write Policies:

- Write-Through: Write to cache and memory simultaneously
- Write-Back: Write to cache only, write to memory on eviction

10. Important Formulas and Diagrams

10.1 Memory Formulas

```
Access Time = Hit Time + Miss Rate × Miss Penalty
Cache Hit Rate = Cache Hits / Total Accesses
Average Memory Access Time = AMAT

Example:
L1 Hit Time = 1 cycle
L1 Miss Rate = 5%
L1 Miss Penalty = 10 cycles
AMAT = 1 + 0.05 × 10 = 1.5 cycles
```

10.2 Virtual Memory Formulas

```
Virtual Address Space = 2^VA_bits
Physical Address Space = 2^PA_bits
Page Size = 2^P_bits

Number of Pages = Virtual Address Space / Page Size
Page Table Size = Number of Pages × PTE Size
```

10.3 DMA Transfer Time

```
Transfer Time = (Data Size / Transfer Rate)
CPU Overhead = Setup Time + (Number of Transfers × Interrupt Time)
DMA Efficiency = Transfer Time / (Transfer Time + CPU Overhead)

Example:
Data Size = 1024 bytes
Transfer Rate = 1 MB/s
Transfer Time = 1024 / 1,000,000 = 1.024 ms
Setup Time = 1 µs
Interrupt Time = 1 µs
CPU Overhead = 2 µs
Efficiency = 1024 / (1024 + 2) = 99.8%
```

11. Practice Problems

Problem 1: Cache Memory

Cache design with:

- 16KB cache
- 64 bytes block size
- 4-way set associative

```
Cache Size = 16KB = 16384 bytes
Block Size = 64 bytes
Set Size = 4 blocks

Number of Blocks = 16384 / 64 = 256 blocks
Number of Sets = 256 / 4 = 64 sets
Set Index Bits = log2(64) = 6 bits
Block Offset Bits = log2(64) = 6 bits
Tag Bits = 32 - 6 - 6 = 20 bits
```

Problem 2: Virtual Memory

Virtual memory with:

- 32-bit virtual address
- 4KB page size
- 24-bit physical address

```
Page Size = 4KB = 2^12
Virtual Page Number Bits = 32 - 12 = 20 bits
Physical Page Number Bits = 24 - 12 = 12 bits
Number of Virtual Pages = 2^20 = 1,048,576 pages
Number of Physical Frames = 2^12 = 4,096 frames
Page Table Entry Size = 12 bits (physical frame number)
Page Table Size = 1,048,576 × (12/8) = 1.57 MB
```

Problem 3: DMA Transfer

Calculate DMA transfer time for:

- 4KB data
- 100 MB/s transfer rate
- 2 µs setup and interrupt time

```
Data Size = 4KB = 4096 bytes
Transfer Rate = 100 MB/s = 100,000,000 bytes/s
Transfer Time = 4096 / 100,000,000 = 40.96 µs
Setup/Interrupt Overhead = 2 µs
Total Time = 40.96 + 2 = 42.96 µs
Efficiency = 40.96 / 42.96 = 95.3%
```

12. Key Takeaways

✅ I/O interface bridges CPU-peripheral speed mismatch
✅ Asynchronous transfer uses handshaking for synchronization
✅ Interrupt-driven I/O is more efficient than programmed I/O
✅ DMA transfers data directly between I/O and memory
✅ Priority interrupts handle multiple devices effectively
✅ I/O Processor offloads I/O management from CPU
✅ Memory hierarchy balances speed, cost, and capacity
✅ Cache memory reduces average memory access time
✅ Virtual memory simulates larger memory space
✅ Associative memory enables content-based search

These comprehensive notes cover Module 5 of Computer Architecture. Focus on understanding I/O interfaces, memory hierarchy, and data transfer modes. Practice DMA calculations and cache memory design problems for better understanding.