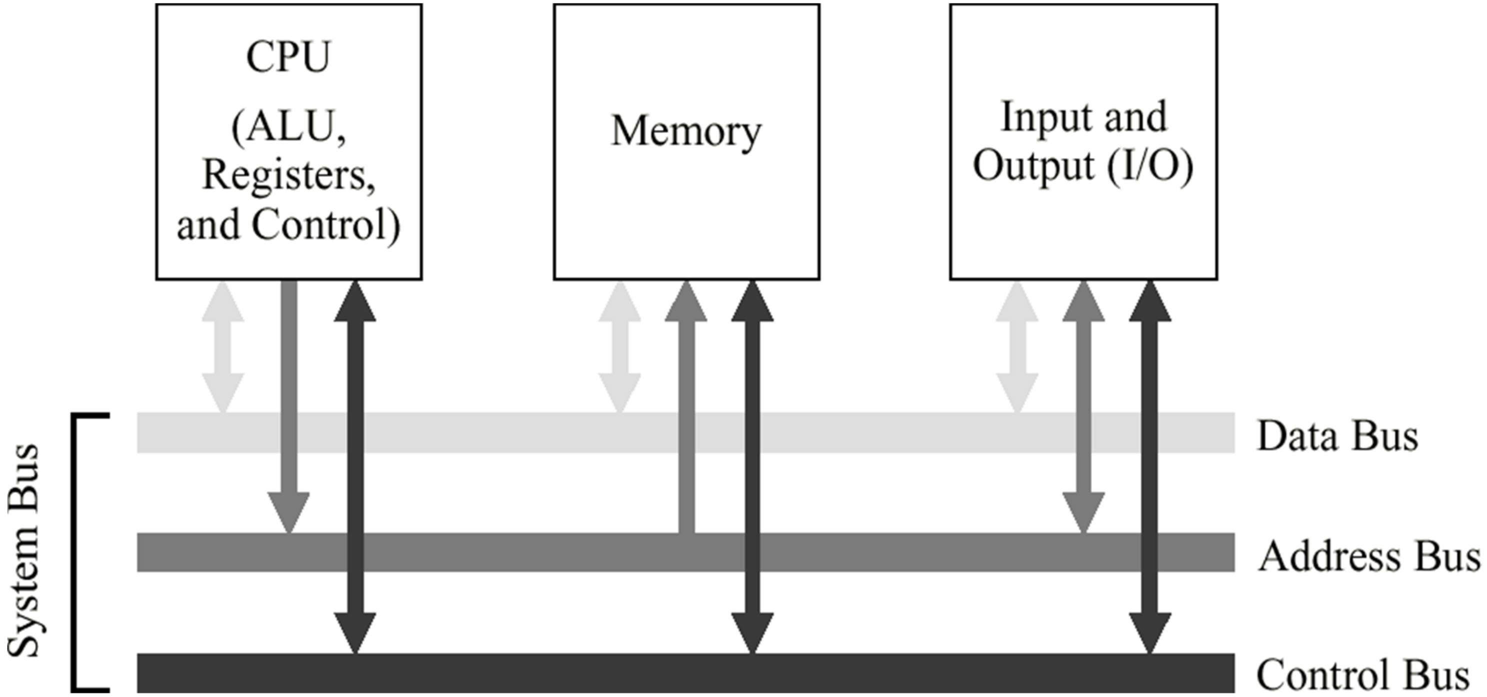
External Memory Interfacing

- System Bus Model
- Memory Mapped I/O
- Direct Memory Addressing (DMA)

The System Bus Model



The CPU must communicate with memory and I/O devices. There are two ways that this can be achieved. The **Mem/$\overline{\textbf{IO}}$** signal, if present, is used to distinguish between the two.

Address Bus

- $\mathbf{A_{N-1}}$ down to $\mathbf{A_0}$ e.g. **A(15:0)** for a 16 bit bus.
- The address bus can be thought of as a binary word with $\mathbf{A_{N-1}}$ being the msb and $\mathbf{A_0}$ being the lsb of the word.
- The state of the N lines (1 or 0) allows the representation of 2^N unique numbers or addresses.

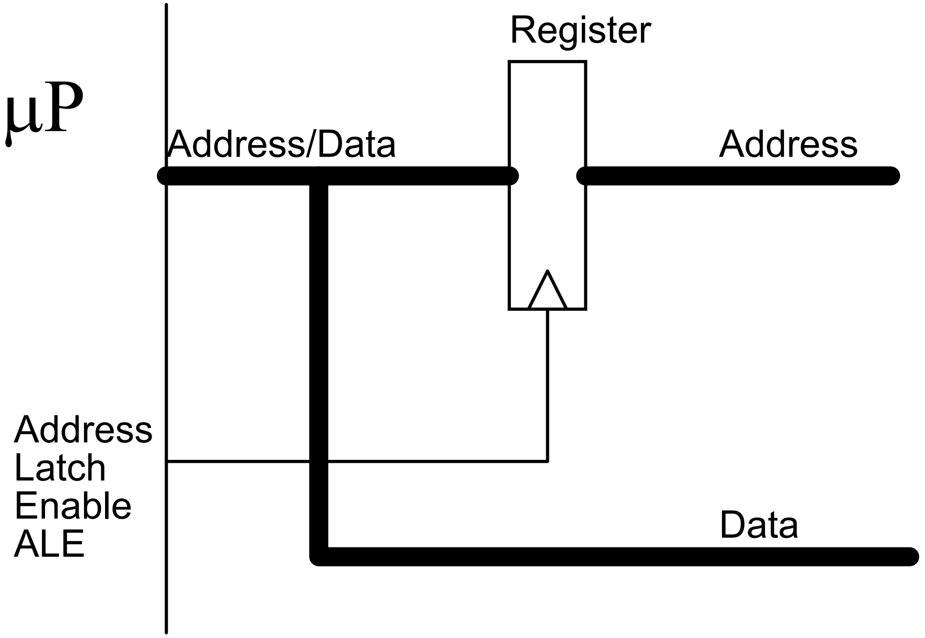
N=16 allows 2^{16} unique locations or *memory locations* to be accessed

Data Bus

- bi-directional because it must allow data to be stored or retrieved.
- M bit word where $\mathbf{D_{M-1}}$ is the msb and $\mathbf{D_0}$ is the lsb.
- Typical data bus widths are 4, 8, 16, 32, and 64.

Multiplexed Address Data Buses

- need to limit pins on a microprocessor package
- address and data buses can be multiplexed on to the same physical pins
- an external latch is used to hold the address valid throughout the memory cycle:

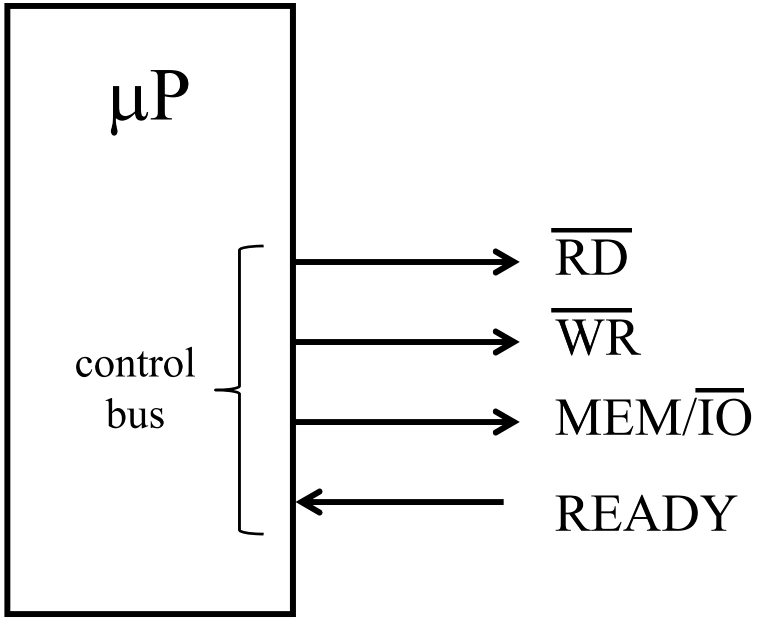


Control Bus

Need to tell the external system (e.g. memory) when the address and data buses contain information and what to do with it.

This bus provides a sequence of events to allow the transfer of data.

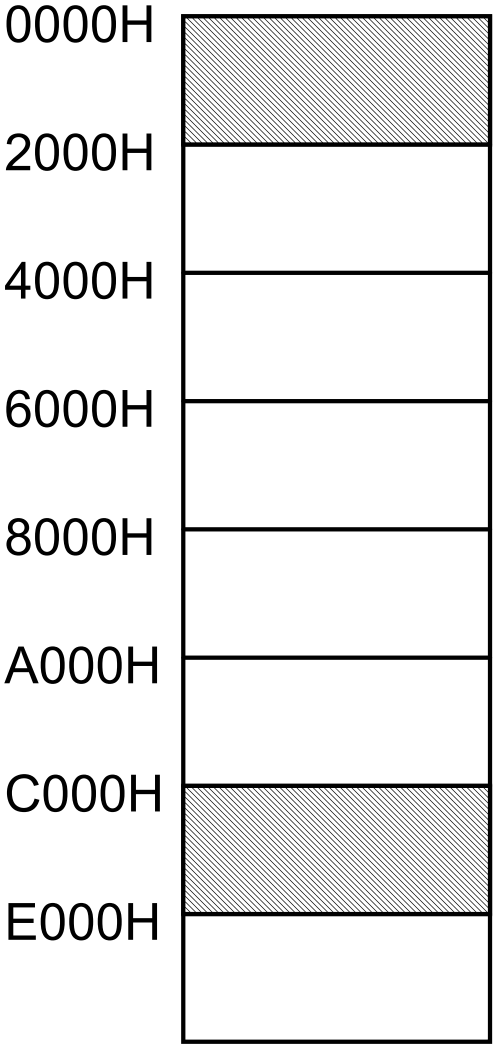
An example of a control bus:



Memory Map

This accessible range of memory locations is often known as the memory or ***address space*** of the microprocessor and the way in which the memory is sub-divided according to use is often called the ***memory map***.

During the boot process, a memory map is passed from firmware to the Operating System.



Memory Mapped I/O

There are two ways that the CPU can communicate with peripheral devices.

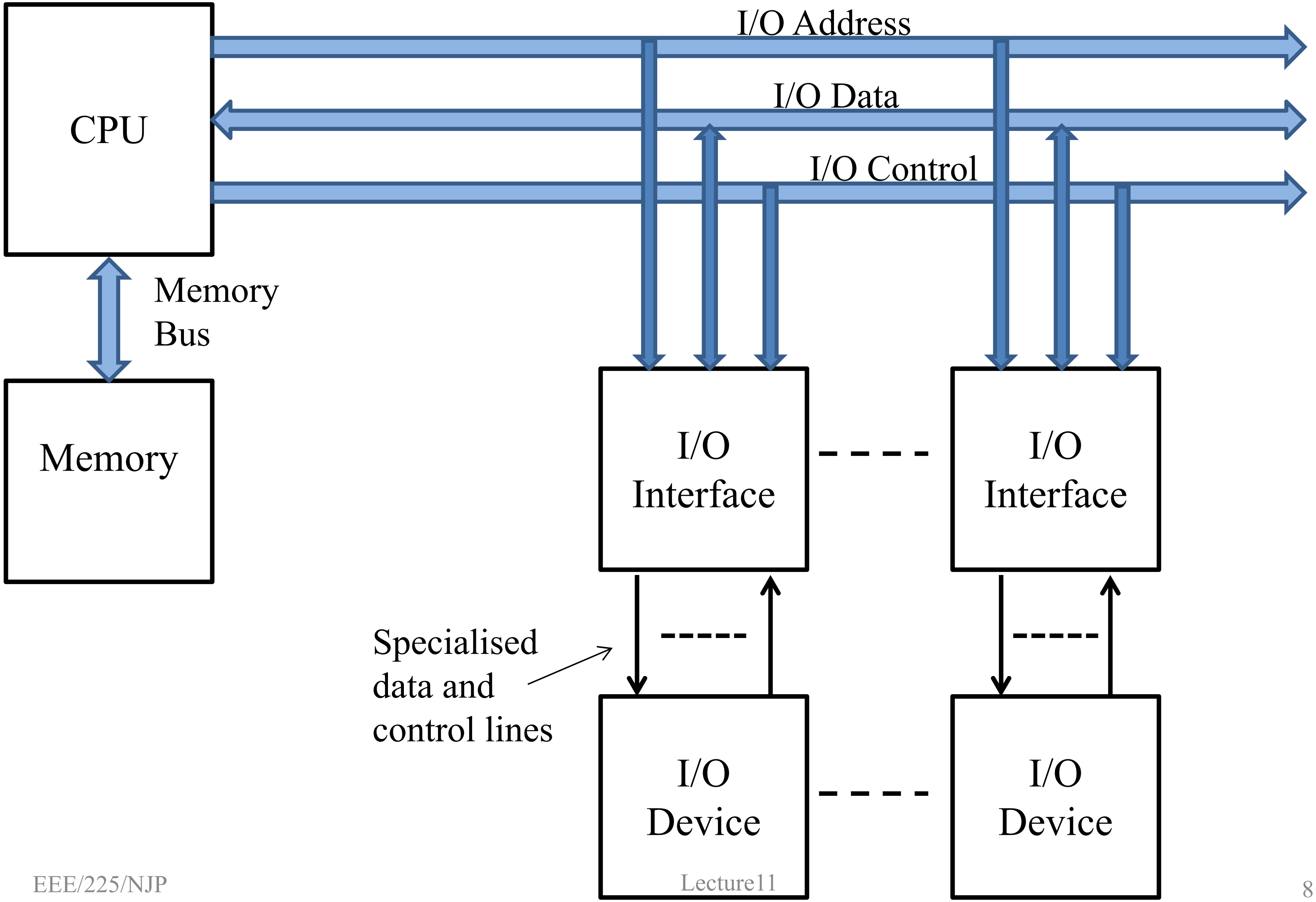
Isolated I/O

- Separate address space for I/O operations.
- Dedicated assembly language instructions **(IN** , **OUT**) for data transfers

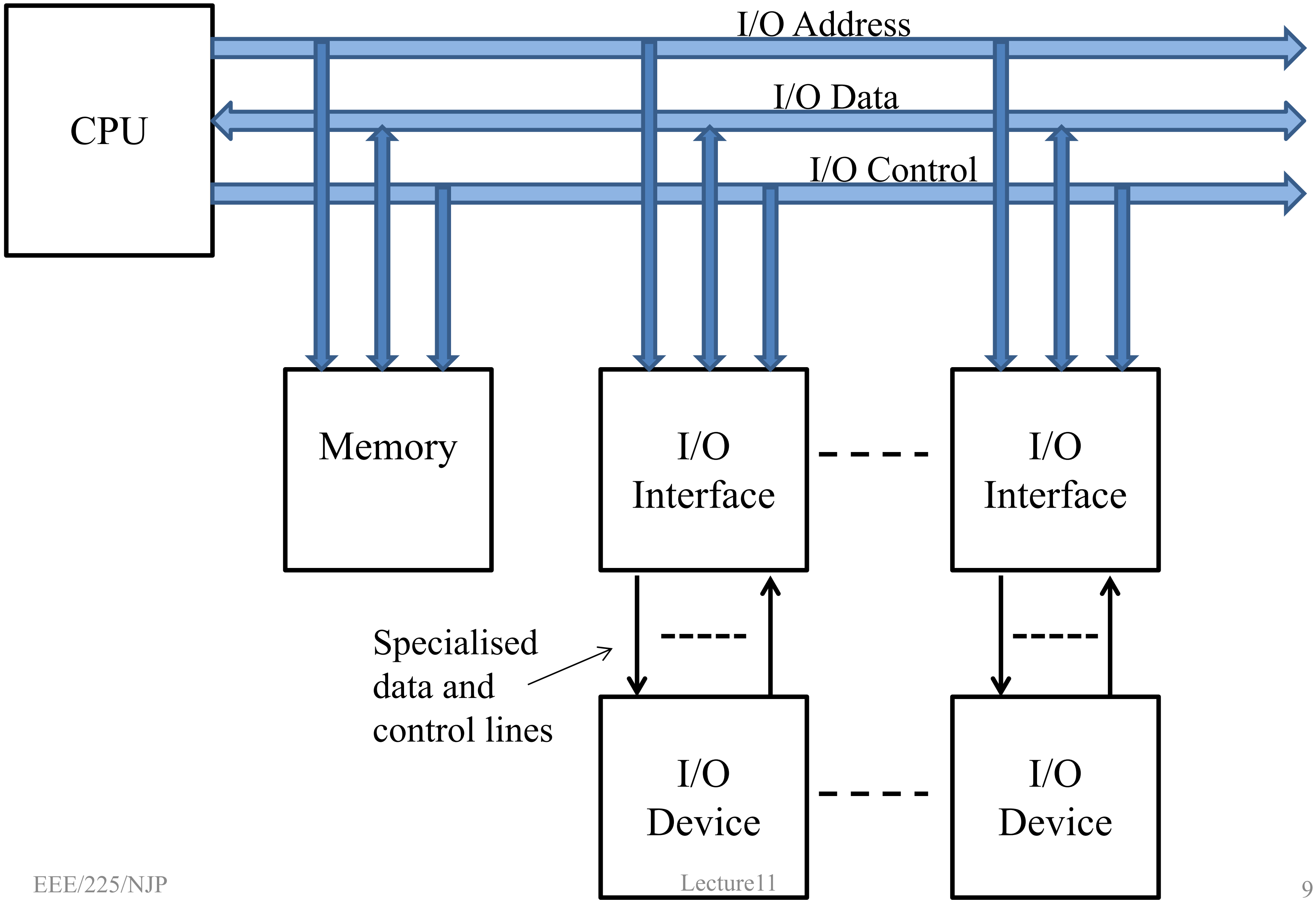
Memory Mapped I/O

- Memory and I/O use the same address space
- Same instruction set to access memory and I/O
- Registers of I/O devices are mapped to the memory address space

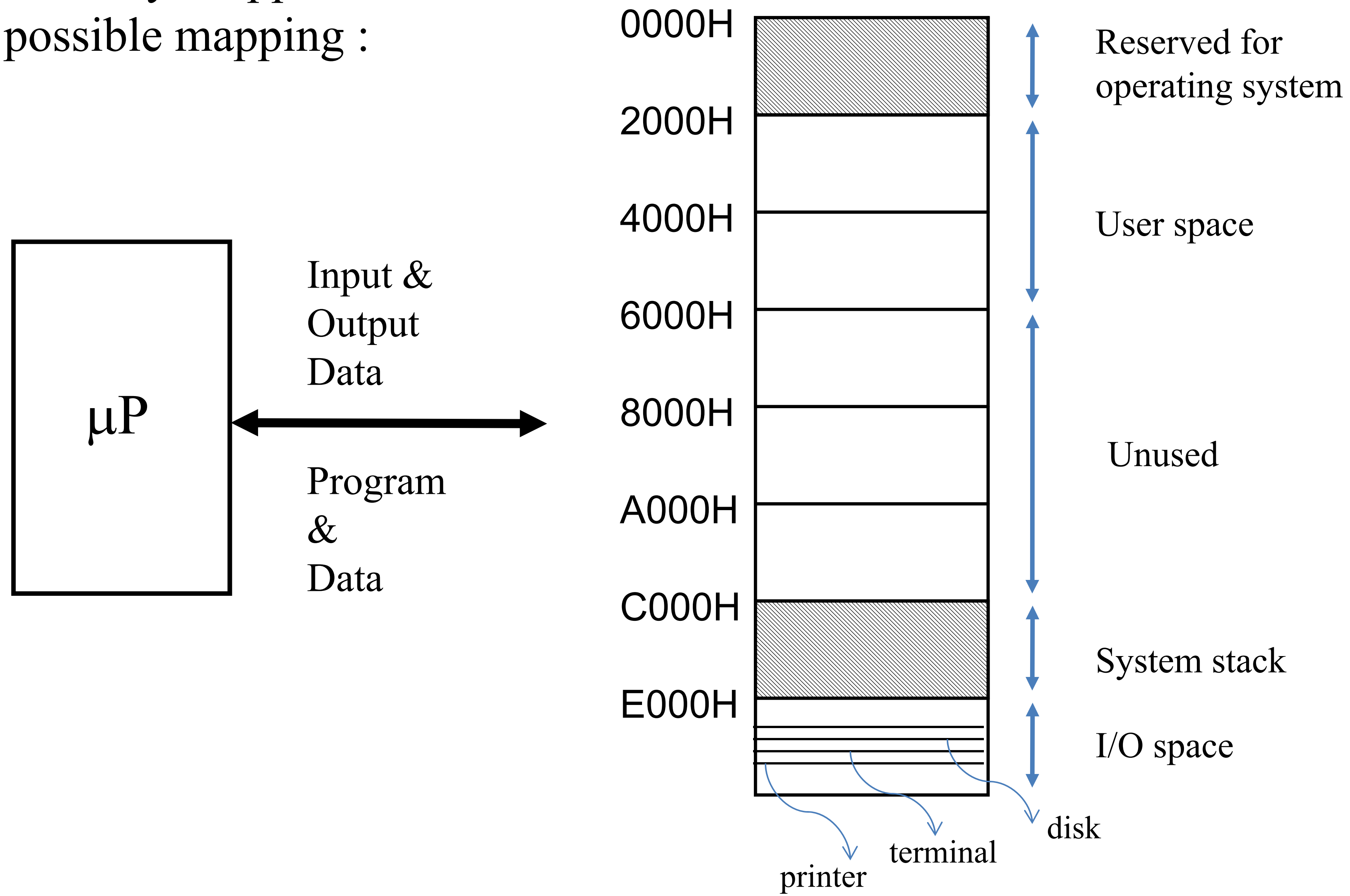
Isolated I/O Structure



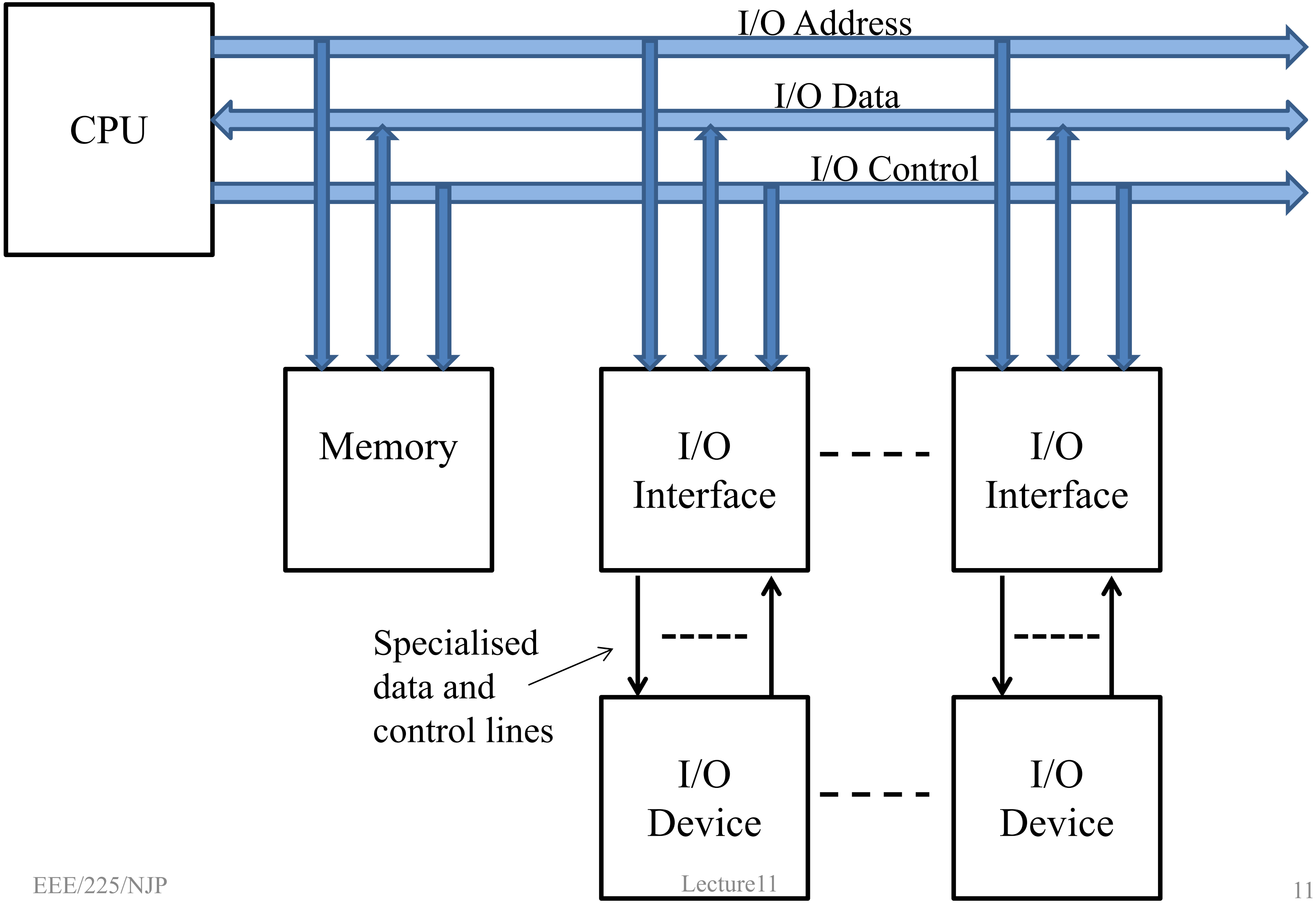
Memory-mapped I/O Structure



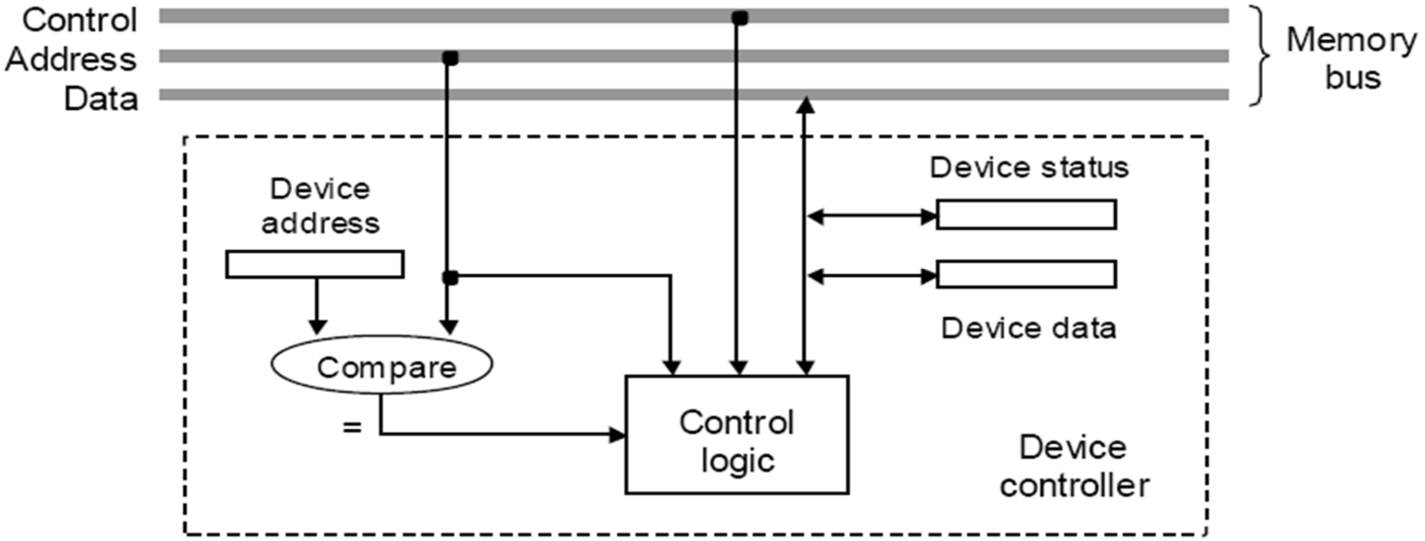
Memory Mapped I/O possible mapping :



Memory-mapped I/O Structure



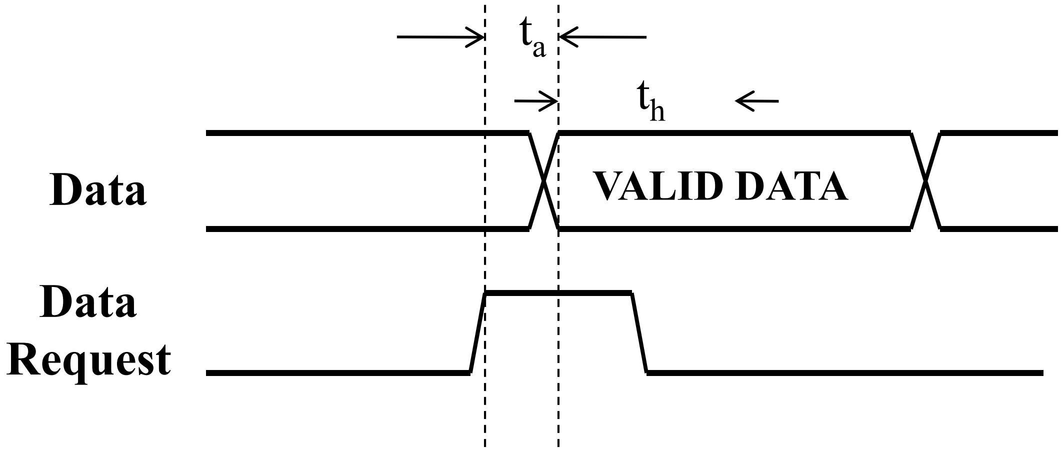
I/O Controller Interface



- Provides a consistent interface to the processor
- Status registers indicate the state of the device
- Processor can examine the status register to check on the transfer or device can interrupt the processor when ready
- Data registers are used for the transfer of data

Asynchronous Data Transfer

Destination Initiated Transfer

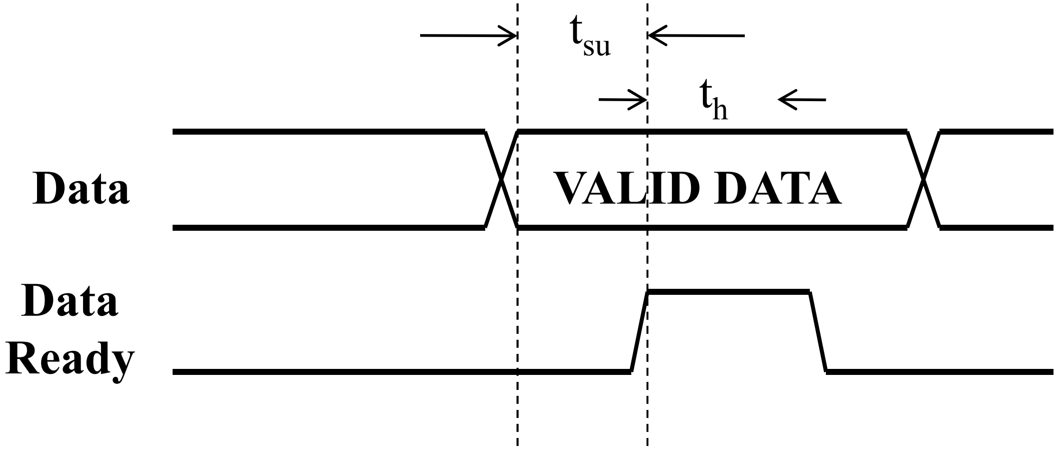


❑ Timing Constraints

- A data access time, t_a
- A minimum data hold time, t_h

1. Destination sends ‘Data Request’ control signal
2. After some delay, the source presents valid data
3. The destination reads the data – perhaps on the falling edge of Data Request
4. The source removes the data

Source Initiated Transfer



❑ Timing Constraints

- A minimum data setup time, t_{su}
- A minimum hold time, t_h

1. Source outputs valid data
2. After some delay, the source asserts a ‘Data Ready’ control signal
3. The destination reads the data – perhaps on the rising edge of Data Ready
4. The source de-asserts Data Ready
5. Source removes valid data

Handshaking

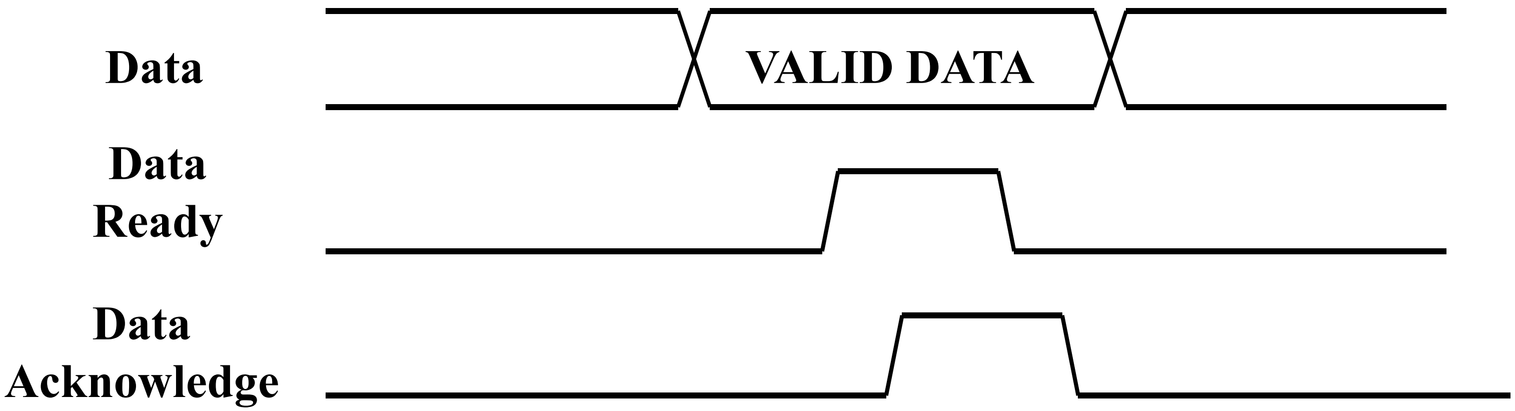
The two previous transfers are known as strobing and will work as long as the responses of both the source and destination are predictable. (i.e. deterministic)

If the response time is unpredictable, there is no way of knowing whether the data transfer has been completed properly.

To ensure proper operation a technique known as *handshaking* is used.

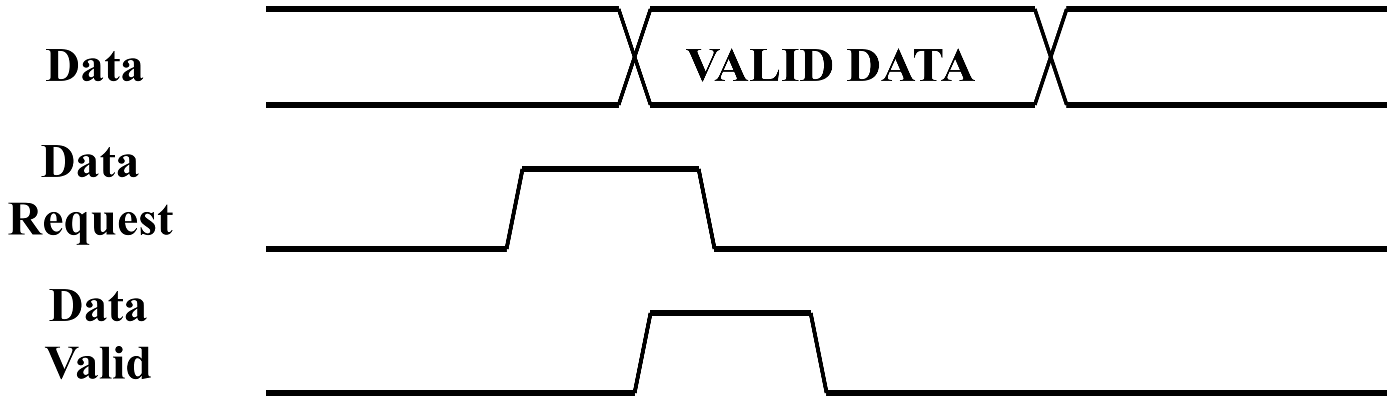
A second control signal is added from the end that did not initiate the transfer.

Source Initiated Transfer with Handshaking



1. Source asserts Data Ready and presents valid data
2. Destination reads the data (perhaps on the rising edge of Data Acknowledge) and asserts Data Acknowledge
3. Source removes data

Destination Initiated Transfer with Handshaking



1. Destination asserts Data Request
2. Source responds with valid data
3. Source asserts Data Valid to say it has successfully produced the data
4. The destination reads the data, perhaps on the rising edge of Data Valid
5. Source removes data

Direct Memory Access (DMA)

Within computer systems, there is frequently a need to transfer significant-sized blocks of data.

Consider how this is done using the CPU:

1. Copy from I/O space to register
2. Copy from register to a location in memory

For example, a Z80 μP would take about 29 clock cycles

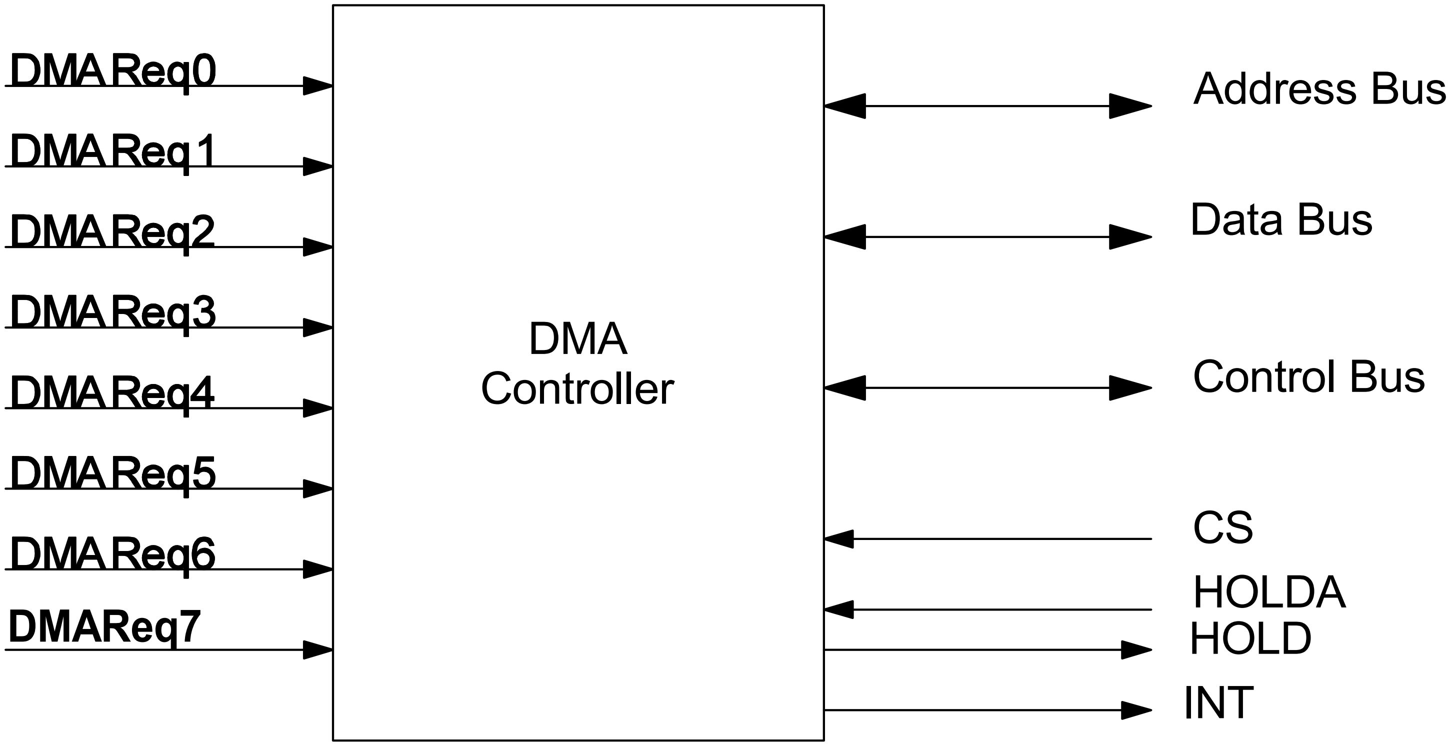
More than 75% of this time is consumed by processor overheads

Very inefficient for transferring blocks of data

- DMA allows an ‘external agent’ to have direct access to a μP’s memory (and I/O)
- Generally used to speed-up block data moves
    - Memory to I/O
    - I/O to Memory
    - Memory to Memory
- The μP cannot access its own memory when DMA is taking place (it can still be processing internally)

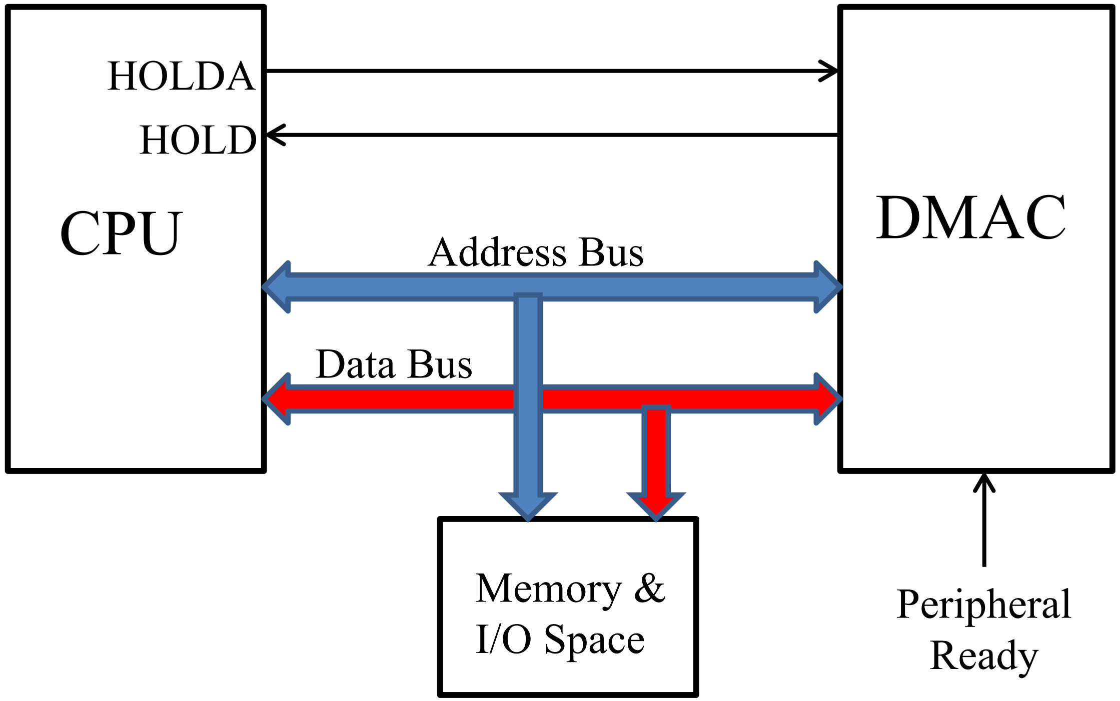
Who is the External Agent?

- DMA Controller (part of the μP chip-set):



- Similar to the Interrupt Controller - simple request lines as input, complexity handled by the DMA Controller

Basic Operation



1. DMAC receives READY signal from peripheral and asserts HOLD
2. When convenient, the CPU grants direct memory access and asserts HOLDA. The CPU also tristates its data & address bus connections and control lines
3. The DMAC transfers data over system bus
4. The DMAC de-asserts HOLD , the CPU de-asserts HOLDA and regains bus control

Modes of Use

- **Byte-at-a-time (Cycle Stealing)**

  The DMAC gains control of the memory bus, transfers a single byte (or word) and then returns control to the CPU.

  - Only one byte transferred during each DMA period
  - Can be inserted at the end of instructions (minimal effect on overall performance)
  - Minimises effect on CPU, limited data transfer rate

- **Burst Mode**

  The DMAC continues to transfer data until the data ready signal of the peripheral is de-asserted whereupon it relinquishes bus control to the CPU.

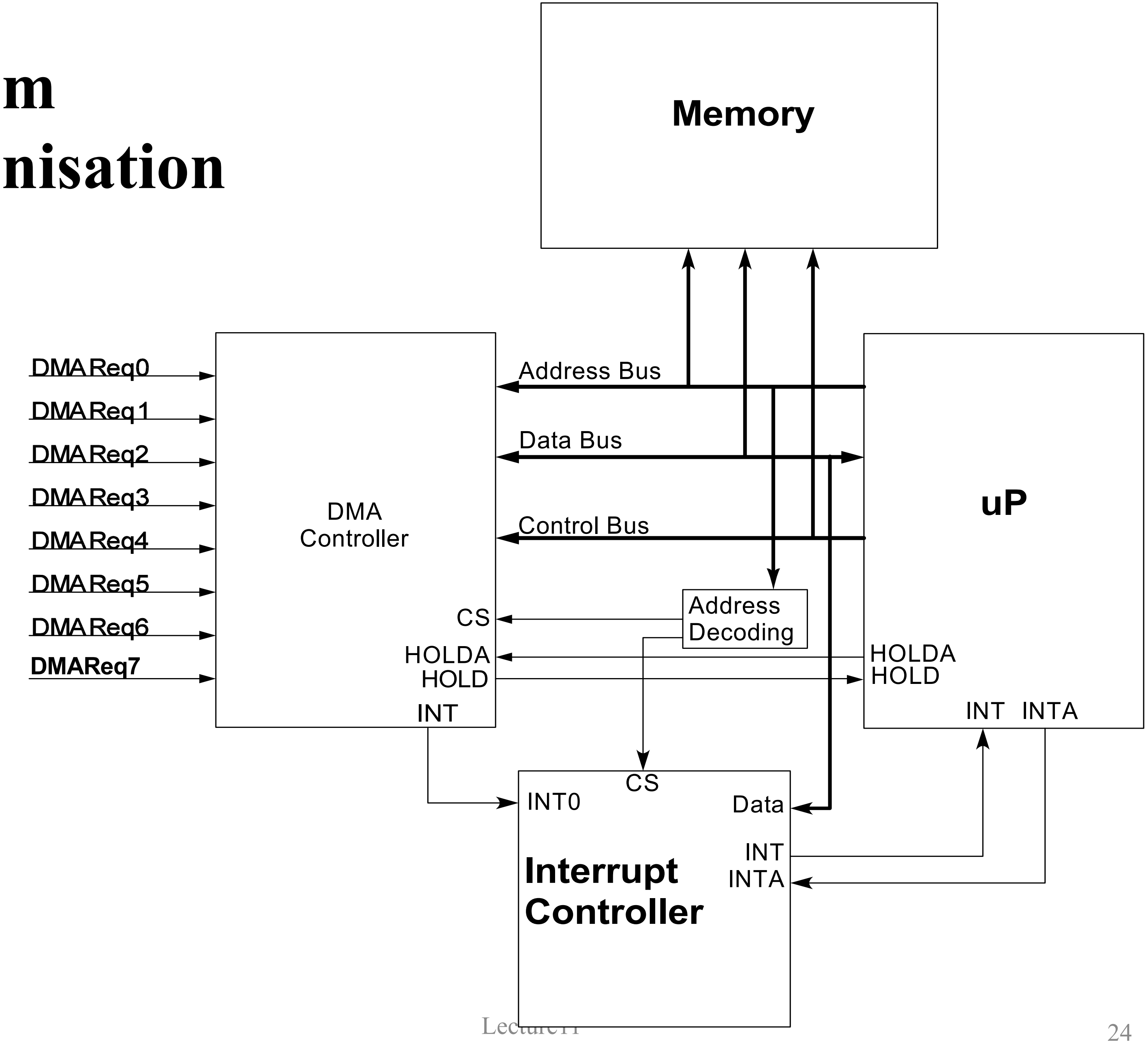
  - Many data may be transferred during each DMA period
  - Moderate transfer rate, moderate impact on CPU performance

- **Continuous Mode**

  DMAC continues to hold memory bus, even if Device Ready signal is de-asserted until the specified transfer is completed.

  - Major impact on CPU efficiency but good transfer rate

System Organisation


Memory
DMAReq0
DMAReq1
DMAReq2
DMAReq3
DMAReq4
DMAReq5
DMAReq6
DMAReq7
DMA Controller
CS
HOLDA
HOLD
INT
Address Bus
Data Bus
Control Bus
Address Decoding
uP
HOLDA
HOLD
INT
INTA
CS
INT0
Data
INT
INTA
Interrupt Controller